Opal-RT RT-XSG toolbox User Guide

RTXSG-UG-11-01

I M A G I N A T I O N

T O

R E A L - T I M E

OPAL-RT
1751 Richardson, suite 2525
Montréal, QC, Canada H3K 1G6
Phone: +1-514-935-2323 • Fax: +1-514-935-4994
www.opal-rt.com

TABLE of CONTENTS

CHAPTER 1: INTRODUCTION

About the Opal-RT RT-XSG toolbox 1
Key Features 1
Hardware description language (HDL) and fixed-point numbering . . . 2
Simulink 2
Organization of this Guide 2
Conventions 3

CHAPTER 2: REQUIREMENTS

Software requirements 5

CHAPTER 3: HARDWARE DESIGN USING THE RT-XSG TOOLBOX

Field-Programmable Gate Arrays (FPGAs) 7
RT-XSG-compatible softwares 7
Matlab/Simulink® 7
Xilinx Integrated Software Environment (ISE®) Design Suite 7
Opal-RT Real-Time LABoratory (RT-LAB®) 7
Introduction to the RT-XSG hardware I/O interfaces 8
RT-XSG FPGA model creation paradigm 8

CHAPTER 4: BUILDING MODELS WITH RT-XSG

System generator for DSP toolbox 11
Gateways 11
Target platform and configuration file version selection 12
Building a RT-LAB-compatible RT-XSG model 14
Augmented Dword 33-bit data vectors 19
Offline simulation of a design 19
Configuration file generation 19
Target platform configuration 20
RT-XSG models invoked from within an RT-LAB model 20
Standalone RT-XSG models 22

CHAPTER 5: TROUBLESHOOTING

Test example models and demos 23
Physical resource shortage 23

APPENDIX A: RT-XSG SIMULINK LIBRARY REFERENCE MANUAL

Opal-RT FPGA Synthesis Manager 26
Version 28
Synchronization pulse train generator 30
DataIN 32
DataOUT 34

© 2009 Opal-RT Technologies Inc.

TABLE of CONTENTS

op_cosin 36
op_trisin 38
OpXsgManager 40
OpSGxPCManager 43
OpXSGscope 46
XSGscopeCmd 49

© 2009 Opal-RT Technologies Inc.

© 2007 Opal-RT Technologies Inc. All rights reserved for all countries.

Information in this document is subject to change without notice, and does not represent a commitment on the part of OPAL-RT Technologies. The software and associated files described in this document are furnished under a license agreement, and can only be used or copied in accordance with the terms of the agreement. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information and retrieval systems, for any purpose other than the purchaser's personal use, without express written permission of OPAL-RT Technologies Incorporated.

Documents and information relating to or associated with OPAL-RT products, business, or activities, including but not limited to financial information; data or statements; trade secrets; product research and development; existing and future product designs and performance specifications; marketing plans or techniques, client lists, computer programs, processes, and know-how that have been clearly identified and properly marked by OPAL-RT as "proprietary information," trade secrets, or company confidential information. The information must have been developed by OPAL-RT and is not made available to the public without the express consent of OPAL-RT or its legal counsel.

ARTEMIS, RT-EVENTS, RT-LAB and DINAMO are trademarks of Opal-RT Technologies, Inc. MATLAB, Simulink, Real-Time Workshop and SimPowerSystem are trademarks of The Mathworks, Inc. LabVIEW is a trademark of National Instruments, Inc. QNX is a trademark of QNX Software Systems Ltd. All other brand and product names are trademarks or service marks of their respective holders and are hereby acknowledged.

We have done our best to ensure that the material found in this publication is both useful and accurate. However, please be aware that errors may exist in this publication, and that neither the authors nor OPAL-RT Technologies make any guarantees concerning the accuracy of the information found here or in the use to which it may be put.

Published in Canada

Contact Us

For additional information you may contact the Customer Support team at Opal-RT at the following coordinates:

Tool-Free (US and Canada)	1-877-935-2323 (08:30-17:30 EST)
Phone	1-514-935-2323
Fax	1-514-935-4994
E-mail	support@opal-rt.com info@opal-rt.com sales@opal-rt.com
Mail	1751 Richardson Street Suite 2525 Montreal, Quebec H3K 1G6
Web	www.opal-rt.com

1 Introduction

1.1 About the Opal-RT RT-XSG toolbox

RT-XSG is a toolbox developed by Opal-RT Technologies inc. It can be invoked from within the Matlab/Simulink or Xilinx Integrated Software Environment (ISE®) Design Suite environments. It can be used in standalone mode in order to provide the configuration data for one of the supported reconfigurable platforms. The tool can also be invoked from within the RT-LAB® environment to provide the user with state-of-the-art solutions for advanced FPGA-accelerated real-time and/or hardware-in-the-loop system simulation.

The user has the freedom to generate a custom, application specific model to be implemented onto the FPGA device. Opal-RT provides signal conditioning and conversion modules that can be attached to the custom model for real-time, hardware-in-the-loop data processing. RT-XSG provides a convenient, Simulink-based way to build the user model. Nevertheless, a user with appropriate knowledge has the possibility to configure the system with an all-VHDL user model. The RT-XSG toolbox brings FPGA-based cosimulation more straightforward by managing automatically the configuration file generation according to the specific processing algorithm to be iplemented on the target platform. It also manages the configuration of the platform, along with the transfer of high-bandwidth data between RT-LAB simulation models and the user-defined custom system running on the FPGA, designed using the RT-XSG Blockset.

1.2 Key Features

Reconfigurability

The supported platform FPGA devices can be configured exactly as required by the user, not just with the board manufacturer default configuration. Integration with Simulink and the System Generator for DSP toolbox from Xilinx allows the transfer of Simulink submodels to the FPGA processor for distributed processing.

In addition, standard and user-developed functions can be stored on the on-board Flash memory for instant start-up. RT-LAB-compatible platforms can be remotely configured using a network-based utility. Additionally, all RT-XSG supported standalone products are configurable on-the-fly using a JTAG connection and the device vendor programming software.

Performance

All of our supported products enable update rates of 100 MHz, providing the capability to perform time-stamped capture and generation of digital events for high precision switching of items such as PWM I/O signaling up to very high frequencies, as I/O scheduling is performed directly on the board. OP5300 family of conversion and conditioning modules provides real-time access to interface I/O signals.

Channel Density

Our supported products let the user configure the I/O interfaces to the FPGA computational node according to its needs. The channel density for each of the supported platform is indicated in the user guide of each specific board (refer to Section 1.3 below).

,Intended Audience and Required Skills and Knowledge

The intended user of the Opal-RT RT-XSG Toolbox is a R&D, algorithm or Test Engineer that needs a reconfigurable, very-high-speed, portable and low-cost processing unit with good analog and/or digital I/O capabilities.

1.2.1 Hardware description language (HDL) and fixed-point numbering

With the help of Xilinx's System Generator for DSP Blockset, only minimal programmable logic technical knowledge is needed to use the RT-XSG-supported platforms. This blockset is used to translate a Simulink design built using particular library blocks into HDL. This translated design is used by Opal-RT tools to give access to I/O interfaces and debugging facilities.

However, the user should be familiar with the fixed-point numerical format and fixed-point data processing. The use of floating point numbers is very heavily resource consuming into FPGA processing devices and is not suitable in RT-XSG devices as the interface to the conversion modules is in a fixed-point format. A minimal training on FPGA architecture is also recommended.

1.2.2 Simulink

Simulink is a software package developed by the Mathworks that enables modeling, simulation and analysis of dynamic systems. Models are described graphically, following a precise format based on a library of blocks. RT-XSG uses Simulink to define models that will be executed by the reconfigurable platform. It is expected that the user has a clear understanding of Simulink operation, particularly regarding the model definition and simulation parameters.

1.3 Organization of this Guide

This document is the user guide. The topics covered are:

- **Introduction on page 1**- Provides an introduction to simulation and the principles behind the use of the RT-XSG toolbox for Matlab/Simulink.
- **Requirements on page 5** - Software requirements for the use of the RT-XSG toolbox .
- **Hardware Design using the RT-XSG toolbox on page 7** - Desciption of the paradigms behind the RT-XSG environment.
- **Building models with RT-XSG on page 11** - Describes the procedure to develop a RT-XSG-compatible model.
- **Troubleshooting on page 23** - Useful topics to resolve RT-XSG problems.

In addition to this guide, the user is invited to each supported board User Guide for information on platform-specific features:

- Opal-RT OP5130 reconfigurable platform;
- Xilinx ML50x family of Virtex-5-based platforms.

1.4 Conventions

Opal-RT guides use the following conventions:

Table 1: General and Typographical Conventions

THIS CONVENTION	INDICATES
Bold	User interface elements, text that must be typed exactly as shown.
Note:	Emphasizes or supplements parts of the text. You can disregard the information in a note and still complete a task.
Warning:	Describes an action that must be avoided or followed to obtain desired results.
Recommendation:	Describes an action that you may or may not follow and still complete a task.
`Code`	Sampel code.
Italics	Reference work titles.
Blue Text	Cross-references (internal or external) or hypertext links.

Requirements

2

2.1 Software requirements

The RT-XSG toolbox needs the following softwares in order to be able to generate a programming file for the reconfigurable device and to program the platform:

Minimal configuration (all-VHDL projects):

- Operating system:
  - Microsoft Windows XP (32/64-bit versions) or
  - Linux Red Hat Enterprise Linux 4 WS (32/64-bit) or
  - Linux Red Hat Enterprise Linux Desktop 5 (32/64-bit) or
  - Linux SUSE Linux Enterprise 10 (SLED) or Server (SLES) (32 and 64 bit);
- Xilinx ISE design suite v10.1.03 with IP Update 3 or later[1].

Recommended configuration (with Matlab/Simulink RT-XSG support):

- Microsoft Windows XP (32-bit version);
- Xilinx ISE design suite v10.1.03 with IP Update 3 or later (See footnote 1.). Xilinx ISE design suite v7.1i can be used with the OP5130 target platform only;
- Xilinx System Generator for DSP v10.1.03 or later (See footnote 1.);
- Matlab R2007b or R2008a. Matlab R14SP1 and Matlab R14SP2 can be used with the OP5130 target platform.

1.Xilinx ISE Design Suite, IP and System Generator for DSP should always correspond to the latest available update. In particular, compatibility issues require the installed release of each component to match (e.g. ISE Design Suite 10.1.03 with IP Update 3 and System Generator 10.1.03, or any later matching release of all the subcomponents). Updating one of the Xilinx subcomponents is likely to require an update of all other Xilinx tools and libraries to ensure full software compatibility.

Hardware Design using the RT-XSG toolbox

3

3.1 Field-Programmable Gate Arrays (FPGAs)

An FPGA is a programmable logic semiconductor device. Depending on the device model, it includes a certain number of programmable logic blocks and built-in functions. Many devices, including all Opal-RT-supported devices, are fully reconfigurable, enabling the user to sequentially perform any number of custom processing on the target platform.

3.2 RT-XSG-compatible softwares

3.2.1 Matlab/Simulink®

MATLAB is a technical computing software package that integrates programming, calculation and visualization. MATLAB also includes Simulink; this software package is discussed below. As RT-LAB and RT-XSG work in conjunction with this environment to define models, the user must be familiar with aspects of MATLAB as related to Simulink.

Simulink is a software package that enables modeling, simulation and analysis of dynamic systems. Models are described graphically, following a precise format based on a library of blocks. RT-XSG uses Simulink to define models that will be converted into configuration data for the targeted platform. It is expected that you have a clear understanding of Simulink operation, particularly regarding model definition and the model various simulation parameters.

3.2.2 Xilinx Integrated Software Environment (ISE®) Design Suite

Xilinx is one of the world major FPGA vendor. The ISE Design Suite is a complete set of tools designed by Xilinx, inc. to access, manage and generate the configuration data for their FPGAs. From within Matlab/Simulink, the System Generator for DSP toolbox, also designed by Xilinx inc., gives access to a block set suited for implementation on an FPGA. It is assumed that the reader is familiar with the Xilinx System Generator for DSP toolbox. Please refer to the Xilinx System Generator for DSP User Guide and introductory tutorials for more information on this toolbox.

Although the user does not have to access them manually, many other components from the ISE Design suite are indirectly called from both the System Generator for DSP and RT-XSG toolboxes. Please refer to the ISE Design Suite documentation for information on each specific component.

Some of the supported platforms enable the creation of VHDL-only model descriptions. This configuration does not require Matlab/Simulink, as the configuration data generation is performed from within the ISE Design Suite Project Navigator. Detailed information on the use of this feature can be found in the specific platform RT-XSG documentation (see section 1.3).

3.2.3 Opal-RT Real-Time LABoratory (RT-LAB®)

RT-LAB™ is a distributed real-time platform that facilitates the design process for engineering systems by taking engineers from Simulink or SystemBuild dynamic models to real-time with hardware-in-the-loop simulations, in a very short time, at a low cost. Its scalability allows the developer to add compute-power where and when needed. It is flexible enough to be applied to the most complex simulation and control problem, whether it is for real-time hardware-in-the-loop applications or for speeding up model execution, control and test.

RT-LAB provides tools for running simulations of highly complex models on a network of distributed run-time targets, communicating via ultra low-latency technologies, in order to achieve the required performance. In addition, RT-LAB's modular design enables the delivery of economical systems by supplying only the modules needed by the application in order to minimize computational requirements and meet customers price targets. This is essential for high-volume embedded applications.

Formerly an integrated component of RT-LAB, the RT-XSG toolbox incorporates features to communicate at very high speed with a RT-LAB model running in real-time.

3.3 Introduction to the RT-XSG hardware I/O interfaces

A general data processing block diagram is illustrated in Figure 1. The role of the RT-XSG toolbox is to provide the user with all the facilities necessary to feed the custom processing block with appropriate data, and to send the generated outputs to an appropriate target. In Figure 1, these roles are symbolized by the two bold arrows.

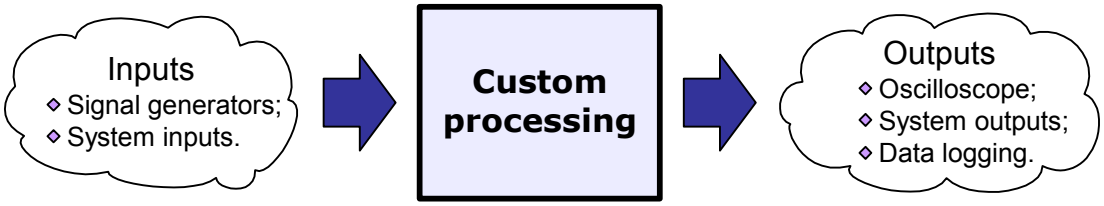

Figure 1:General data processing block diagram.

The ‘Input’ and ‘Output’ can be implemented as needed by the application. As examples, consider the four following cases:

- Direct input and monitoring devices, as external signal generators and oscilloscopes, respectively;
- In a hardware-in-the-loop simulation, outputs are used outside the box to generate the inputs directly by the hardware under test;
- In a FPGA-accelerated simulation, as when RT-XSG is used in conjunction with RT-LAB, the ‘Custom processing’ block is used to offload part of the processing from the software processor onto the FPGA board. In this case, inputs come from the processor, and the outputs loop back to the same software model;
- Any combination of the above.

The type of input/output channel configuration is application specific. Nevertheless, the maximum channels count is platform-dependent, and is indicated in the specific platform RT-XSG documentation (see section 1.3).

3.4 RT-XSG FPGA model creation paradigm

In Figure 1, the ‘Custom Processing’ block is designed by the user. It is often refered to as the “User model”. For simplicity purposes, some structural and technical features are transparent from the user when working with the RT-XSG toolbox. Specifically, the User model signals are attached to a “base configuration”, which is the top-level hierarchical entity of the reprogrammable device and is invisible to the user. The base configuration serves as an interface from the user model to the outside world (i.e. the components and ports on the target platform outside the programmable device). It manages signal routing and I/O configuration compatibility with the user model, high-speed bidirectional

communication with any RT-LAB model, along with the LCD user interface for platforms that incorporate such feature.

The RT-XSG toolbox provides a series of block libraries that give access to a variety of analog and digital I/O interfaces, along with blocks that enable the transfer of digital signals to and from a RT-LAB simulation model in real-time. The toolbox facilitates the interface management so that the user can concentrate on the algorithmic processing part of the design.

Note: A RT-XSG model is usually designed from within the Matlab/Simulink environment. Blocks fron the RT-XSG libraries incorporate System Generator blocks under their mask. Moreover, the FPGA User description model must be built using ONLY blocks from the System Generator for DSP Blockset. It is advised to pass through the System Generator for DSP tutorials before starting to use the RT-XSG toolbox.

Building models with RT-XSG

4

This chapter covers important topics related to the creation of a RT-XSG Simulink model. It is assumed that the user is already familiar with the System Generator for DSP toolbox.

4.1 System generator for DSP toolbox

Xilinx System Generator for DSP is a toolbox provided by Xilinx that consists in two Simulink simulation libraries[1]. Using the blocks in these libraries and blocks from the RT-XSG library, a user can construct and simulate his own FPGA design, download it to the FPGA chip of the reconfigurable I/O card supported by Opal-RT Technologies, and integrate it in a real-time simulation. Moreover, the System Generator for DSP toolbox allows a user to create and simulate his own FPGA design without the need for knowing traditional HDL languages, all in a Matlab/Simulink environment.

The design can implement DSP algorithms like filters, CORDIC algorithms, PWM generators, waveform generators and much more, and can interface with the I/O cards supported by Opal-RT Technologies.

Note: Simulink designs made using the System Generator for DSP Blockset use intrinsically fixed-point data processing algorithms. Good knowledge of this numbering format is strongly recommended for the designers of Opal-RT RT-XSG models and, more generally, of any design including blocks from the System Generator for DSP Blockset.

4.2 Gateways

The System Generator for DSP toolbox is able to convert a model-based Simulink design into an Hardware Description Language (HDL) file. A programmable device configuration file is then generated from this HDL description. Input and output ports of the model to be implemented on such device are inserted in the Simulink model as “Gateway In” and “Gateway Out” blocks, from the Xilinx Blockset (See Figure 2).

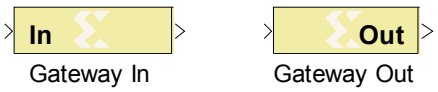

Figure 2:Gateway In and Gateway Out blocks

In an RT-XSG model, the target board is selected by the user in the first designing steps. As the board layout is fixed, the user does not have control on the input and output port definition. The RT-XSG library block sets provide the user with all necessary interface blocks. Although the “Gateway In” and “Gateway Out” blocks are not directly visible by the user on the top hierarchical level of the model, they are still present under the mask of each of these blocks. In general, interface blocks between the User model and the external world show a “blue \ yellow” background pattern, while interface blocks

1.Refer to the System Generator for DSP User Guide for further information and tutorials on how to use the toolbox.

between the user model and a RT-LAB CPU model have a "blue \ turquoise" background pattern. As an example, Figure 3 gives the interface blocks available for a design targeted for the OP5130 board.

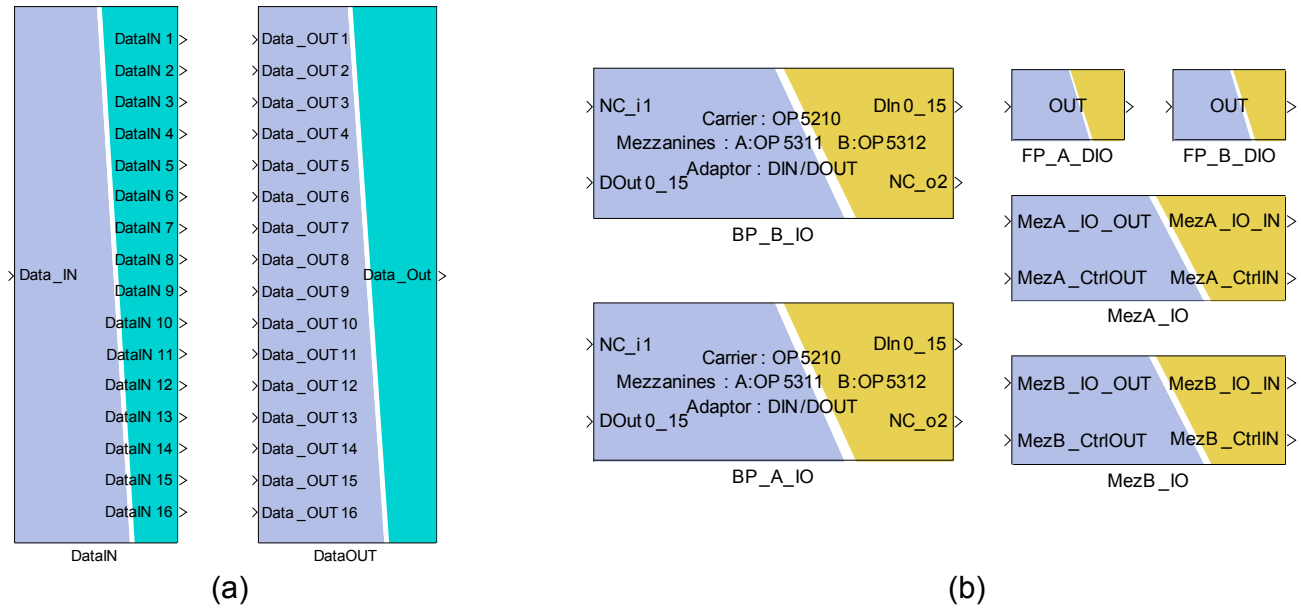

Figure 3:OP5130 interface blocks (a) to the RT-LAB model and (b) to the external world.

As the user does not have control on the physical board layout, no additional gateway should be added by the user other than the ones located inside the RT-XSG library blocks.

4.3 Target platform and configuration file version selection

The target platform is selected from the "Opal-RT FPGA Synthesis Manager" block, located in the *RT-XSG/Tools Blockset*. The target platform is selected by the "FPGA development board" drop-down list (See Figure 4). Many configuration settings are automatically set upon the selection of the target platform.

Note: As the different target platforms have different I/O capabilities, the choice of the interface blocks is strongly dependent on the selected board. Refer to each board documentation for information on the interface blocks compatibility.

SynthesisManager

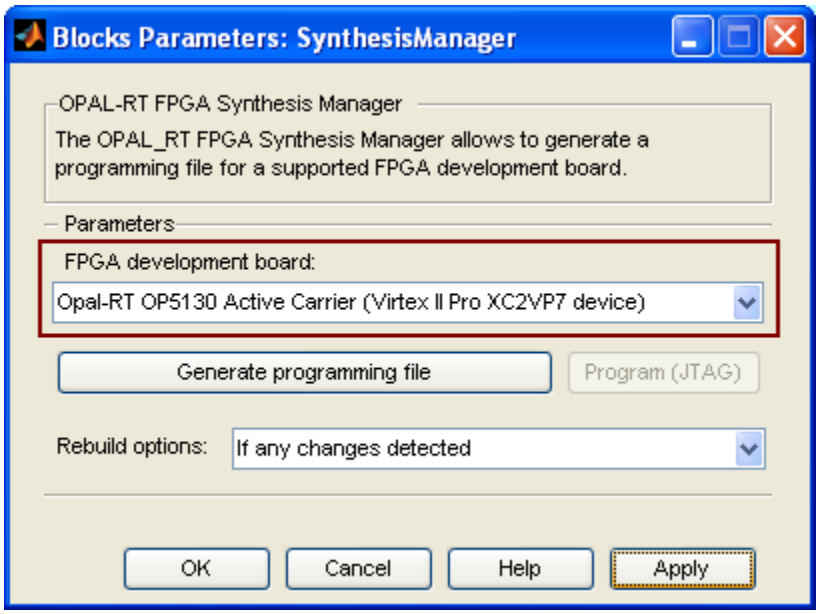

Figure 4:Opal-RT FPGA Synthesis Manager icon and mask. The target platform is selected by the "FPGA development board" drop-down list.

The configuration file version is used to identify the function of any FPGA configuration. From RT-LAB, it is possible to retrieve the configuration file version used to configure any RT-XSG-compatible programmable device. For target platforms with an integrated LCD interface, the version is displayed on the display. The configuration file version is the combination of the release identification number ("Version") and the minor identification number ("Minor ID"). Generally, a single minor identification number is assigned to a specific intended behavior of the FPGA configuration, while the release identification number identify subsequent versions of the same design. Figure 5 shows the icon and mask of the "Version" block, used to set those two identification numbers, located in the *RT-XSG/Common Blockset*.

Version

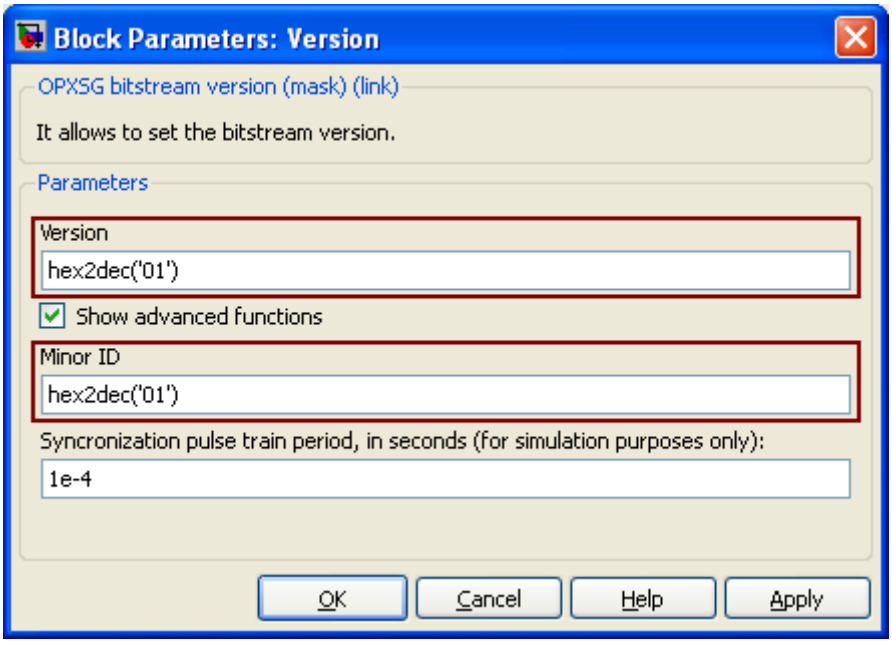

Figure 5:"Version" block icon and mask, used to set the configuration file version identification numbers.

4.4 Building a RT-LAB-compatible RT-XSG model

When designing and RT-XSG-based RT-LAB application, two Simulink models must be created. The first one, hereafter called the "FPGA model", contains the RT-XSG blocks required to build the configuration file to be downloaded in the reconfigurable chip of the target platform.

The second model, the "CPU model", runs on the target node and must contain one interface block, the "OpCtrlReconfigurableIO" block from the *'RT-LAB I/O/Opal-RT/Reconfigurable IOs' Blockset*, in order to manage the communication between the CPU model and the reconfigurable board. This block can be interpreted as a bridge between software and hardware. It communicates and receives in real-time the data samples to and from the OP5130 reconfigurable IO card through the Opal-RT SignalWire communication link.

Example CPU and FPGA models are provided with RT-LAB under folder <RTXSG_ROOT Folder>\Examples\Rt-Lab\ and can be used as a start point for building your specific FPGA design. In this section the "Basic\xsg_demo1" demo model will be studied to demonstrate various characteristics of such models, particularly concerning the fixed-point numbering format, typical to programmable logic devices.

The OpCtrlReconfigurableIO block block can be seen in the example CPU model presented in Figure 6. The Figure 7 shows the mask parameters of this block. In this example, the OpCtrl ReconfigurableIO block is set up with 6 input and 11 output communication ports, thus allowing transfer of six 32-bit words of data to and eleven words from the reconfigurable card at each calculation step. The number of input and output ports of the OpCtrlReconfigurableIO block is configurable between 0 and 16. Note that the width of each inport and outport can also be increased up to 250 words if required by the application using configuration parameters from the DataIN and DataOUT blocks in the FPGA model. The data lines are accessible in the CPU model by multiplexing multiple 32-bit signals on the same Simulink net connected on a single input to the 'Op Ctrl Reconfigurable IO' block, or by demultiplexing multi-dimensional data coming from an output port of the same block.

The "XSG model name" parameter indicates the name of the .mdl file containing the FPGA model. Opening the corresponding model file lets the user design the part of the computation that is to be performed by the reconfigurable board, as illustrated in Figure 8. The model is a mixture of System Generator for DSP library blocks and blocks from the RT-XSG toolbox libraries, which is also built using blocks from the System Generator library. The RT-XSG 'DataIN' and 'DataOUT' library blocks control data transfers to and from the CPU model, through the SignalWire link.

Sample model for accessing a Reconfigurable Active Carrier board

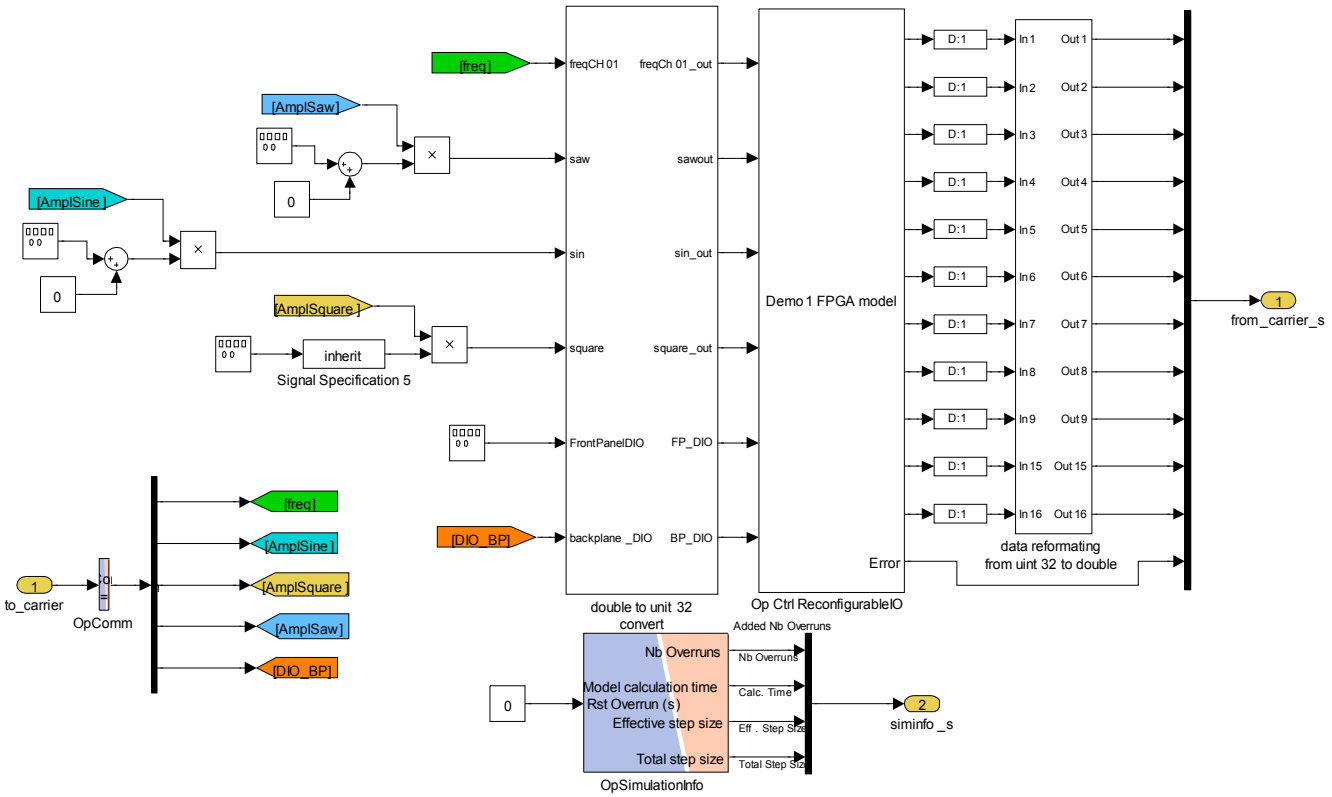

Figure 6:An example of RT-LAB master subsystem controlling a reconfigurable I/O board with the ‘Op Ctrl Reconfigurable IO’ block.

The DataIN and DataOUT blocks each provide 16 ports of 33 bits (see Section 4.5) in the Xilinx UFix format. Data coming from the DataIN block is updated at the rate of the CPU model simulation time step. A synchronization pulse train whose period is equal to the specific CPU application time step is available as a signal named “ModelSync” (available by using a Simulink “From” block). This rate is adjusted to the actual CPU model step size at the start of the CPU model execution. This rate is typically ranges from tens to hundreds of microseconds, which is much larger than the FPGA clock period (usually 10ns). On the DataOUT side, again, the sample period is 10ns on the input ports of the block but data samples are sent to the CPU model at the CPU model rate.

The yellow RT-XSG blocks represent the different I/O channels available on the reconfigurable board. In this example, the design gives access, via the MezA_IO and MezB_IO blocks, to two 52-pin mezzanine connectors of the OP5130 card (for connecting one 16-channel D/A and one 16-channel A/D mezzanine boards, the OP5330 and OP5340 respectively). It also gives access to the front panel digital lines (FP_A_DIO and FP_B_DIO) and to the board backplane connectors (BP_A_DIO and BP_B_DIO), to which digital signal conditioning interface cards or analog conversion modules can be connected.

In this particular example, the input ports of the OpCtrlReconfigurableIO block placed in the CPU model are connected to signal generators that generate samples to be transmitted to the OP5130 at a rate of Ts=200µs. These signal generators create a saw-toothed, a sine and a square waveform. Notice that these signals pass through a subsystem named "double to uint32 convert" that converts the generated double type signals into the uint32 type, as shown in Figure 9(a). This is the only type supported by the OpCtrl ReconfigurableIO for all its inputs or outputs. Along with doing signal type conversion, this subsystem does signal scaling and concatenation. Scaling is necessary before the type conversion so that decimal values are not truncated. In this particular case, the waveform signals are routed to a DAC

interface in the FPGA XSG model which expects the Xilinx Fix16_11 format (refer to the library blocks help files for details on the signaling details), so a "Shift Arithmetic" block shifts the three waveform signals by 11 bits to the left (i.e multiply them by 2^{11}).

Inside the three 'Concatenation' subsystems you will find the type conversion blocks as well as the concatenation logic, as can be seen in Figure 9(b). Concatenation is necessary in this case because the waveform generators are connected to a DAC I/F with 16 bit channels.

Input port In1 represents the lower 16 bits (lowest significant bits, or LSBs) and input port In2 the upper 16 bits (most significant bits, or MSBs) of the concatenated 32-bit word. In this example, both ports are connected to the same source and come out of output port 1, 'saw_out', of the example subsystem of Figure 9(a). This output can be connected to any of the input ports of the OpCtrl ReconfigurableIO block. The *i*-th input port of the OpCtrl ReconfigurableIO block in the CPU model correspond to the DataIN*i* port of the DataIN block in the RT-XSG FPGA model. In this example, the saw_out signal could be connected directly to the OP5330 (bank of digital-to-analog converters) controller, after passing through the Signal Wire link (between the OpCtrl ReconfigurableIO and DataIN blocks). Each of the DAC input port of this block represents two 16 bit concatenated channels, exactly as it was formatted in the RT-LAB model, so no further signal transformation is needed.

Signal concatenation is not required, but it is nonetheless advantageous because it uses the available bandwidth more efficiently.

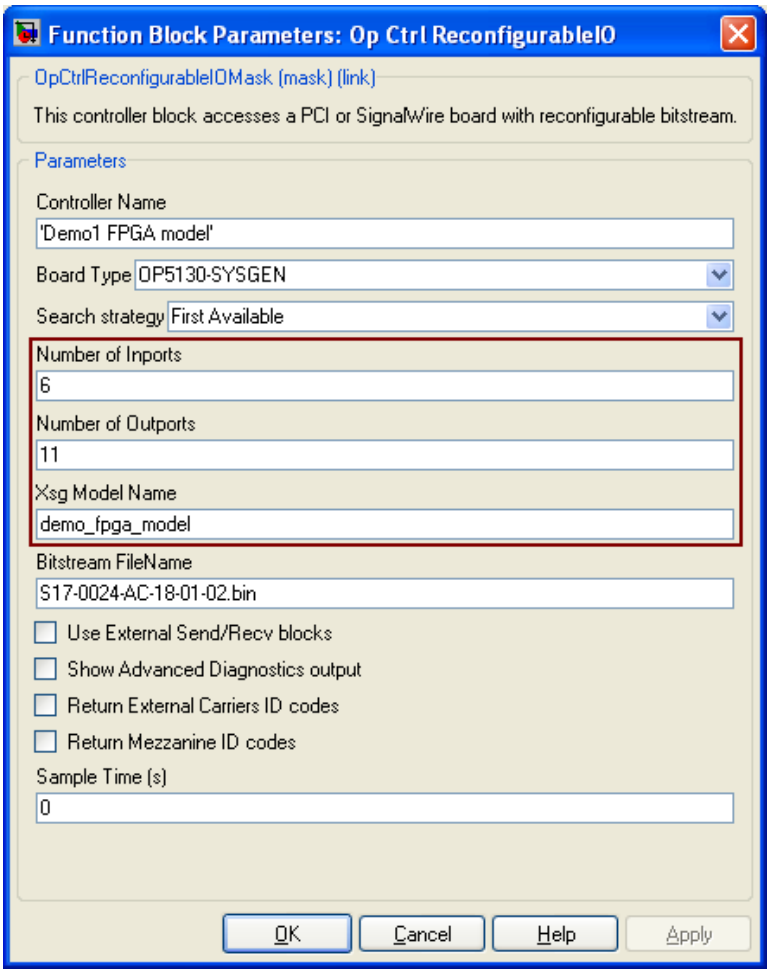

Figure 7:Op Ctrl Reconfigurable IO block mask parameters.

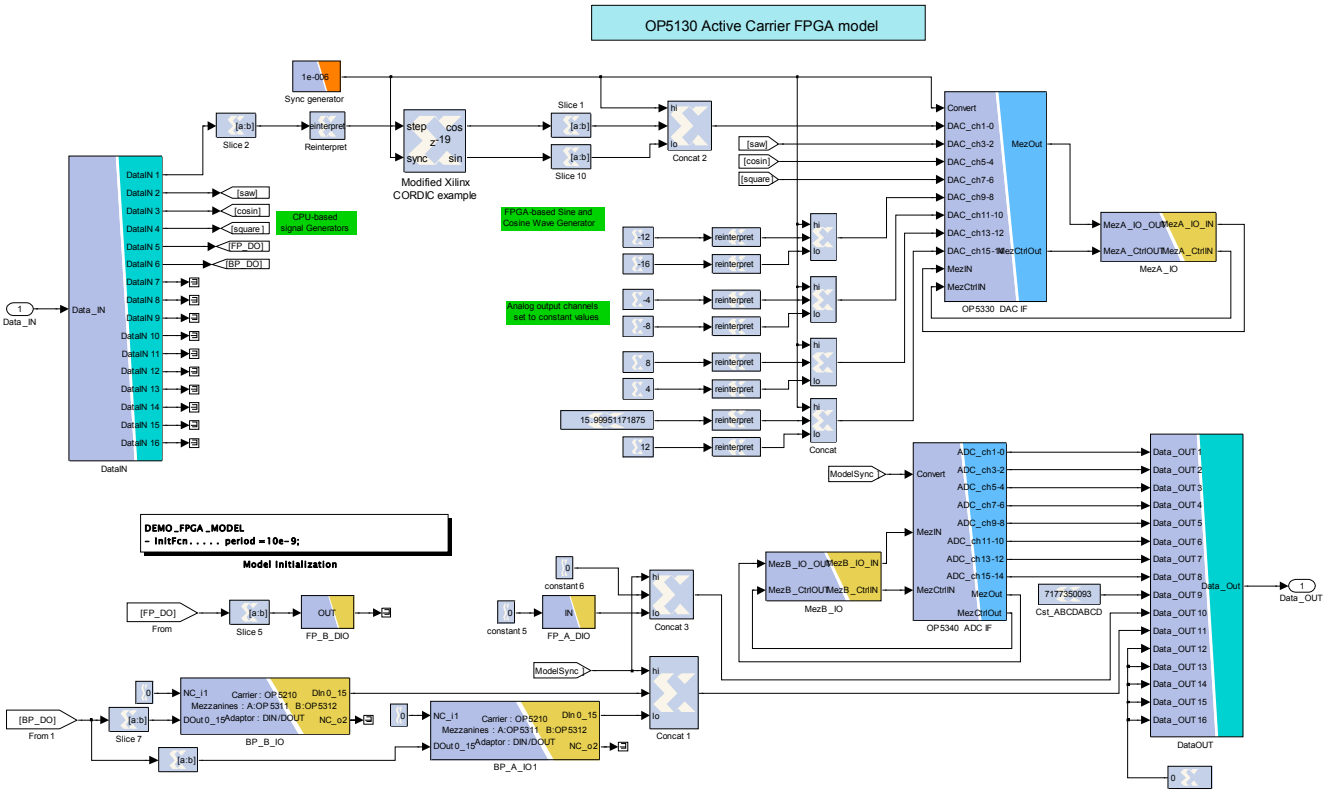

Figure 8:Example OP5130 subsystem of an FPGA model.

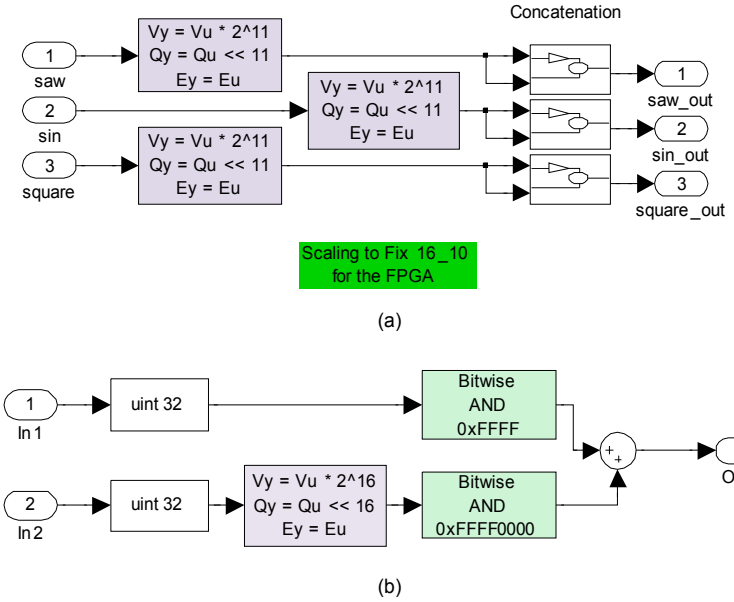

Figure 9:(a)Conversion subsystem connected to the inports of the OpCtrlReconfigurableIO block. (b) Concatenation of two 16-bit data samples into one single 32-bit word to be transmitted to the OP5130.

In the example, the OpCtrl ReconfigurableIO block also receives frequency information on its first input port, which corresponds to the DataIN1 port in the FPGA model. The numerical format of this

information is set to Fix_20_17 in the FPGA model, and thus needs to be transformed in the CPU model as in the case of the waveform signals described above. This data is first scaled by multiplying it by 2^{17} and then converted to a uint32 format. No concatenation is done in this case as the 12 MSBs are unused. Once received by the FPGA model on port DataIN1, the 20 LSBs are extracted from the 32 bit data by a System Generator "Slice" block. Then, a System Generator "Reinterpret" block converts the UFix_20_0 vector output by the slice block into a Fix_20_17 format as expected by the "Modified Xilinx CORDIC example" block. The purpose of this subsystem is to implement a sine and cosine waveform generator using a *CORDIC* algorithm[1]. The frequency of the sine and cosine waveform is modifiable by the target node via the CPU model. The sample rate is controlled by a synchronization pulse train generator block ("Sync Generator") that controls both the rate at which the data samples come out of the "Modified Xilinx CORDIC example" block and the conversion rate of the OP5330 DAC controller block. The pulse train period is set to '1e-006' or 1µs, which is the maximum conversion rate of the OP5330 module.

Note: Implementing the waveform algorithm in the FPGA allowed for much faster sample rates (1 MHz instead of 5 kHz).

Since the output of the CORDIC SINCOS generator is of Fix_20_17 format and the OP5330 DAC controller expects two concatenated Fix_16_10 vectors, it is necessary to extract the 16 MSBs out of the SIN and COS outputs of the CORDIC generator using "Slice" blocks. These two 16-bit signals are then concatenated with the help of a System Generator "Concat" block. The resulting 32-bit vector is now ready to be used by the DAC interface.

The data vectors do be sent by the FPGA model to the CPU model are connected to the "DataOUT" block. Eight of these signals come from an analog-to-digital conversion interface block called "OP5340 ADC IF". This represents the Opal-RT OP5340 16 bit/16 channel analog input card. Each of the outputs of this block represents the concatenation of the 16-bit acquisition data values of two consecutive channels of the card. The Convert port of the ADC I/F is connected to a "From" block with a tag to a ModelSync signal. The ModelSync a reserved signal name which corresponds to a pulse train of 10ns pulse width and a pulse period equal to the sample rate of the CPU target node. By connecting the "ModelSync" pulse train to the Convert port, the ADC I/F will sample at the same rate as the target node. This is different from using a SyncGenerator block as was done in the model for the DAC interface, since using the ModelSync ensures that the sampling is performed in synchronization with the model calculation step, preventing data loss.

The samples available at the DataOUT ports are transmitted to the target node via the SignalWire link once per calculation step. Once the data reaches the target and is placed at the outputs of the "OpCtrl ReconfigurableIO" block, some transformation may be needed in order to extract the samples. The Figure 10 below shows the type of transformation performed in the "data reformatting from uint32 to double" subsystem in the CPU model master subsystem to extract the 16-bit ADC samples and convert them from the uint32 format output from the "OpCtrl ReconfigurableIO" block to the double format. Note that shift arithmetic blocks is used to scale the information properly. Since ADC samples are in the Fix_16_10 format in the FPGA, a shift by 10 bit to the right is necessary in the CPU model.

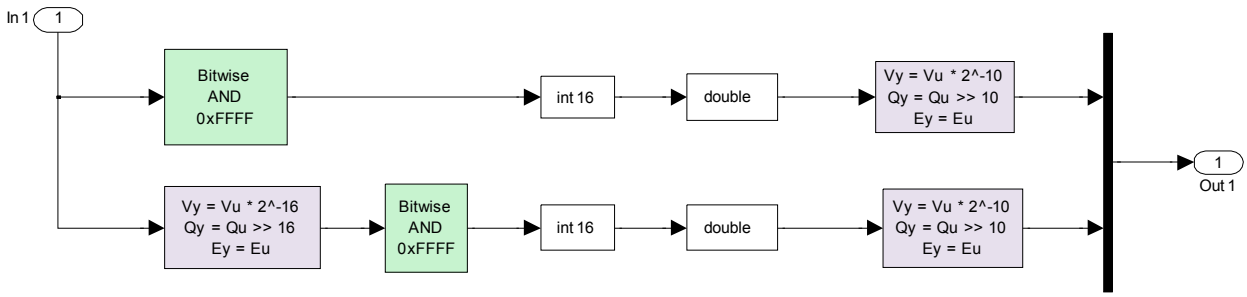

Figure 10:Conversion subsystem connected to one of the output ports of the OpCtrlReconfigurableIO block.

1.Jack E. Volder, *The CORDIC Trigonometric Computing Technique*, IRE Transactions on Electronic Computers, September 1959.

4.5 Augmented Dword 33-bit data vectors

Many RT-XSG blocks communicating 32-bit data vectors (“double words”, or *dwords*) have input and/or output ports 33-bit wide. This format is refered to as an “Augmented Dword”. The 33rd bit (the MSB) is a “valid” bit, used to sample the 32 LSBs by a register or a buffer. Thus, the data output from the DataIN and from the ADC interface blocks can be sampled when the MSB is found to be active. Parallely, the DataOUT and the DAC interface blocks sample the input data when the MSB of the 33-bit inputs is active. A typical application of the 33rd bit of the Augmented Dword to register the data is illustrated by Figure 11.

In consequence, outputs from the first blocks can be fed directly to the inputs of the latter blocks, and the data sampling will be performed correctly. Note that even if the “valid” bit is activated only once for each sample period of each signal of the DataIN and ADC interface blocs, the output data remains unchanged until the next pulse on the 33rd bit of each output vector port.

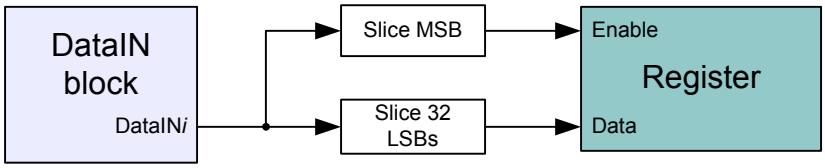

Figure 11:Use of the 33rd bit of the Augmented Dword to register the 32-bit data vector.

4.6 Offline simulation of a design

Offline simulation of a RT-XSG design in conjunction with a RT-LAB model is not yet possible. Offline simulation of a standalone RT-XSG model is possible by following the procedure described in the System Generator for DSP toolbox documentation.

4.7 Configuration file generation

The configuration file generation is performed automatically from the “Opal-RT FPGA Synthesis Manager” block. This block is located in the *RT-XSG/Tools Blockset*, and must always be present in a RT-XSG model, along with the “Version” block, found in the *RT-XSG/Common Blockset*.

The generation is launched by clicking the “Generate programming file” button in the block graphical user interface (GUI) (See Figure 12). The file creation may require a lot of time to complete, depending on the host computer performance, the model complexity[1] and the resources available in the target platform FPGA. Typical configuration file generation time ranges from ten minutes to several hours.

In order to generate the programming file, the following steps must be performed:

- Verify the correctness of the design using the “Update diagram” button from the Simulink toolbar and correct the errors, if any;
- Insert a ‘Opal-RT FPGA Synthesis Manager’ block into your design. In this block GUI, select the appropriate reprogrammable platform from the list and click the “Generate programming file” button.

1.The model complexity in this case is a multidimensional term. One side of a complexity problem is related to the number of basic logical blocks required to perform the processing described by the model. However, the more complex issue of *timing constraints* is of major influence on the time required to generate an FPGA configuration file. In particular, long combinational paths in the desing may induce routing delays in the FPGA nets in the order of the chip clock period. If the propagation of a signal in any combination path in the design exceds the FPGA clock period, the configuration file generation will fail. Routing the desing according to timing constraints specific to each board is an iterative operation, and may require much time to complete.

The following steps are performed during the configuration file generation:

1. The configuration options associated to the specific target platform chosen in the Opal-RT Synthesis Manager block are set. A "System Generator" block from the System Generator for DSP toolbox library is added to the design with the appropriate options. Thus, any manual setting done by the user in this block is not used for the configuration file generation;
2. The RT-XSG model is completed by adding dummy elements to unused inputs and output ports of the model (for example, if any interface block is not present in the original RT-XSG model);
3. All required hardware cores are generated using the Coregen tool from the ISE Design Suite;
4. The "System Generator" block is invoked and the Configuration file generation is performed. A new window is displayed, logging the ongoing process. During this step, several tools from the Xilinx ISE design suite are called successively. If any error occurs during this step, the following files contain the log of the processes until the error occurs:
   - <model folder>\RT-XSG Reports\Synthesis.result:

     Log of the synthesis process, doring which the generated HDL code is compiled and translated into logical equations;
   - <model folder>\RT-XSG Reports\Xflow.result:

     Log of the following processes, during which the synthesized design is converted into elements specific to the targeted FPGA device. Those elements are then placed into the device and routed together according to the specific timing constraints of the target platform. Generation errors, including resource shortage or routing errors, can be found by parsing this file.

   The result of this step is the FPGA confiruration file itself (**.bit*);
5. The target platform Flash memory configuration file is generated from the FPGA configuration. The Flash memory enables the device to reconfigure itself automatically after the system power-on. The format of this file is platform-dependent (**.bin or *.mcs*);
6. Configuration files are copied into the model folder.

Note: For a programming file to be generated, the user must set the "Rebuild option" parameter to 'Always' or 'Only if changes needed'. This requirement is included to prevent unwanted compilations, as this operation can take from several minutes to several hours to complete, depending on the system characteristics.

4.8 Target platform configuration

4.8.1 RT-XSG models invoked from within an RT-LAB model

For RT-XSG models invoked from within an RT-LAB model using an "Op Ctrl Reconfigurable IO" block, the target board is automatically configured when the RT-LAB model is loaded onto the target computer using the "Load" button in the RT-LAB GUI (See Figure 13). Any configuration problem is displayed in the RT-LAB log window. The configuration files are transfered to the target along with the RT-LAB CPU model.

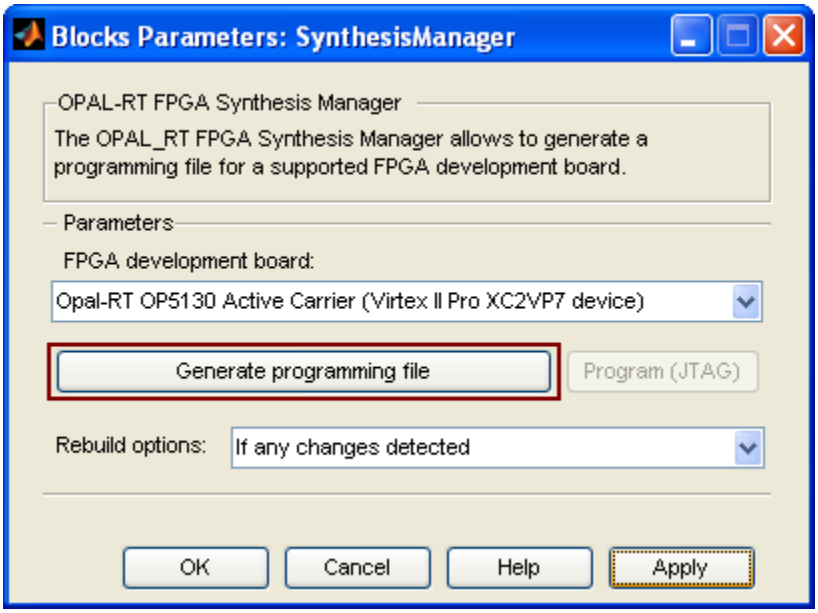

Figure 12:The FPGA configuration file is created by clicking the "Generate programming file" button in the Opal-RT FPGA Synthesis Manager block GUI.

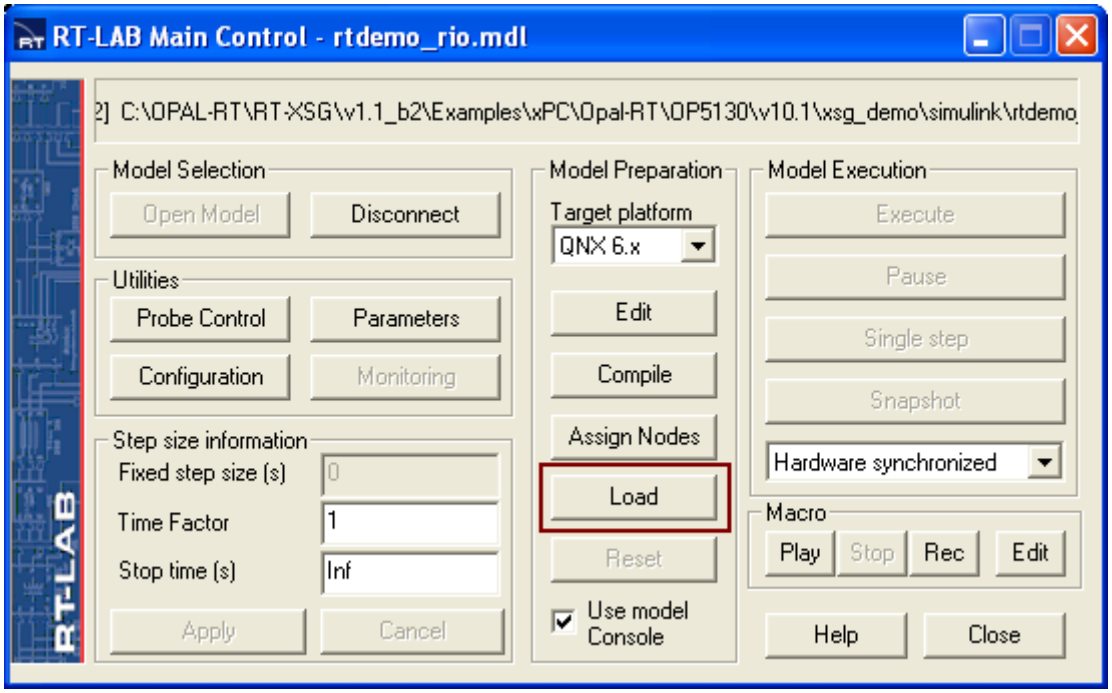

Figure 13:For RT-LAB models, any FPGA board is configured using the Load button in the RT-LAB model.

4.8.2 Standalone RT-XSG models

For standalone RT-XSG models, the FPGA board configuration files are transfered to the target via a JTAG download cable. The configuration is performed using the iMPACT tool from the ISE Design Suite. iMPACT is invoked automatically from the Opal-RT Synthesis Manager block in the Simulink RT-XSG model using the "Program (JTAG)" button (see Figure 14). This button is enabled only is a standalone-mode-compatible board is selected in the GUI.

Before the configuration is performed, the opportunity to save a back-up copy of the current board configuration is given to the user.

Specifically, after the generation of a valid programming file, the target platform is configured by performing the following steps:

- Connect a JTAG platform cable from the host computer to the target platform;
- Power-up the target platform;
- Click the "Program (JTAG)" button from the 'Opal-RT FPGA Synthesis Manager' block GUI (see Figure 14);

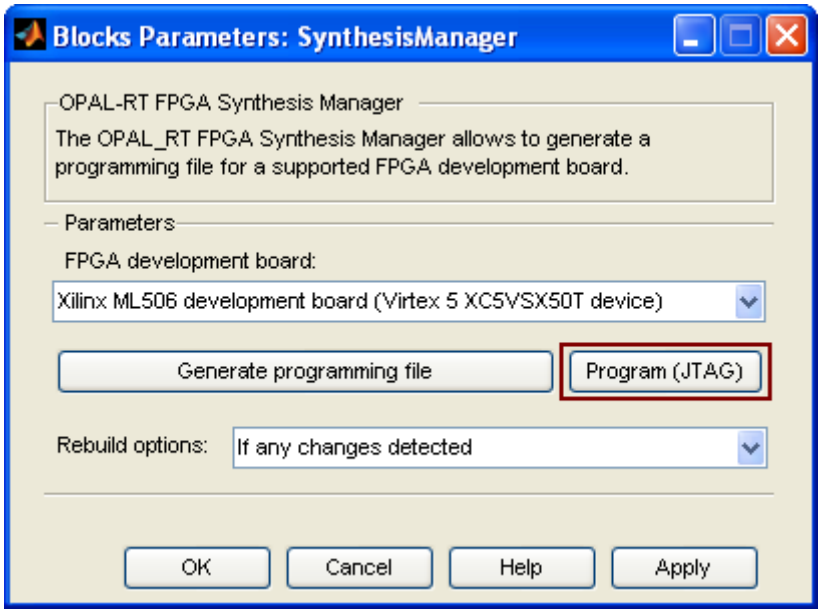

Figure 14:For Standalone-mode models, the configuration is performed using a JTAG download cable by clicking the "Program (JTAG)" button in the Opal-RT FPGA Synthesis Manager GUI.

5 Troubleshooting

5.1 Test example models and demos

Numerous example models can be accessed from Matlab demo browser. To open the example models, type `>> demos` on the Matlab prompt, and run one of the "Opal-RT RT-XSG" demo models. The example models are located in the following folder: <RT_XSG root folder>\Examples.

5.2 Physical resource shortage

Resource management in an FPGA design is a complex science. Even if the target platform includes a dense reconfigurable device, resource may come to an end. In this case, design optimization for resource usage must be performed. The first step to this optimization is to look at the map (or XFlow.results) report.

This file gives informations on which type of resource is insufficient in the device. The designer must find which of the blocks in the design can be optimized to free these resources. In particular, a small decrease applied to memory or DSP block port widths can improve significantly the resource usage efficiency of the entire system.

Parallely, routing problems may occur if long combinational paths are found between sequential elements in the system. Routing fails if the optimizer is not able to ensure that the signal coming out from any sequential element has the time to reach all its target sequential elements between two successive sampling steps. This problem can be resolved by either inserting sequential elements (registers or delay blocks) into the path to resample the data or by decreasing the sampling rate of the signal.

Appendices

RTXSG-UG-11-01

I M A G I N A T I O N

T O

R E A L - T I M E

1751 Richardson, suite 2525
Montréal, QC, Canada H3K 1G6
Phone: +1-514-935-2323 • Fax: +1-514-935-4994
www.opal-rt.com

Opal-RT FPGA Synthesis Manager

Library

RT-XSG/Tools

Block

Synthesis Manager

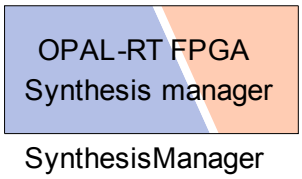

Figure 15:Synthesis Manager block

Mask

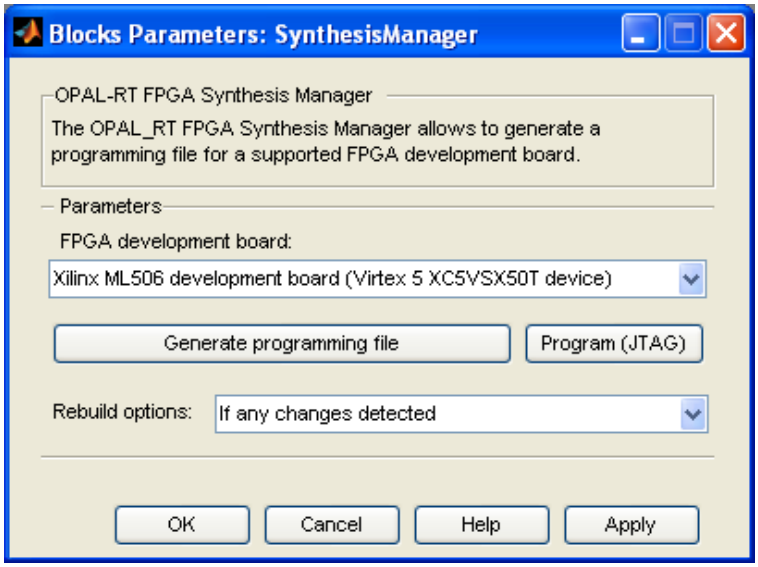

Figure 16:Synthesis Manager graphical user interface.

Description

The FPGA Synthesis Manager is a convenient utility to manage model translation into FPGA-interpretable VDHL code and to integrate this model into the framework of the Opal-RT communication and I/O interfaces *base configuration*. It enables the user to generate a

programming file for the reconfigurable chip of various boards. It also enables the automatic configuration of the board using a JTAG connection.

Refer to the Opal-RT RT-XSG overview[1] and to the Xilinx System Generator for DSP User Guide[2] for more info.

Parameters

FPGA development board: This parameter presents the list of the supported boards for programming file generation.

Generate programming file: Use this button to generate the FPGA programming file corresponding to the current RT-XSG Model.

Program (JTAG): Use this button to program the target FPGA reconfigurable development board. If requested, a new programming file will be generated according to the current model. You will be prompted whether to save a back-up copy of the current FPGA configuration of the reconfiguration board to skip this step.

Rebuild Options: This parameter gives the user three choices:

- **Always:** This choice allows a user to force the compilation of an FPGA configuration file no matter if changes where made or not to the FPGA RT-XSG model.
- **If any changes detected:** Compile the FPGA configuration file only if changes to the FPGA RT-XSG model are detected.
- **Never:** Never generates the FPGA configuration file, no matter changes to the model. This avoids accidental FPGA compilations (such compilations may take several minutes to complete).

Inputs

This block has no inputs.

Outputs

This block has no outputs

Characteristics and Limitations

In the current version of RT-XSG, only standalone target reconfigurable FPGA boards are supported.

Direct Feedthrough	N/A
Discrete sample time	N/A
XHP support	N/A
Work offline	N/A

1.Opal-RT RT-XSG overview: *http://www.opal-rt.com/productsservices/hardwarecomponents/xsg/index.html*
2.Available from Xilinx Website: *http://www.xilinx.com/products/software/sysgen/app_docs/user_guide.htm*

Version

Library

RT-XSG/Common

Block

Version

Figure 17:Version block

Mask

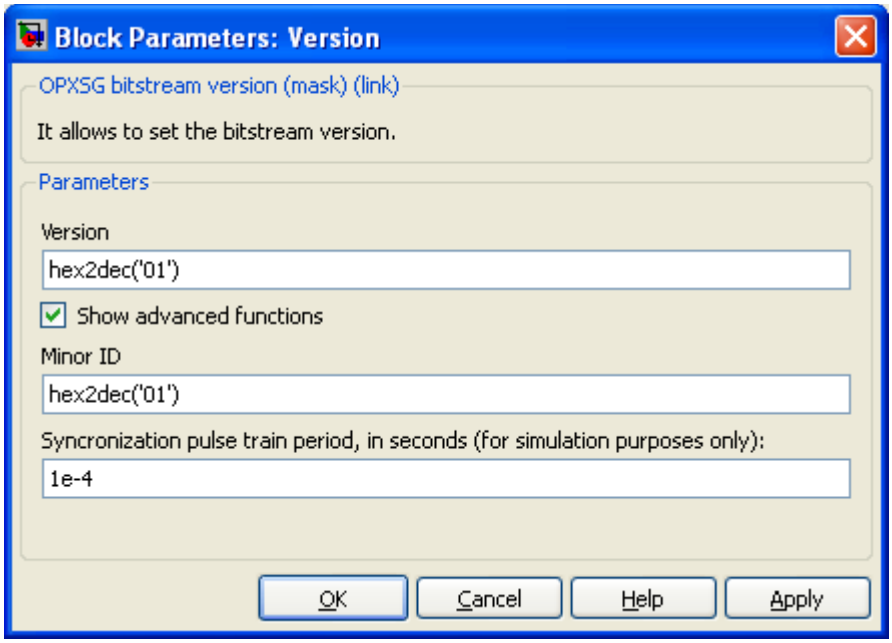

Figure 18:Version mask.

Description

This block allows a user to set the version of the configuration file that will be generated when the RT-XSG model is compiled.

Parameters

Version: This parameter will appear in the last portion of the configuration file name right before the ".mcs". For example, a value of 16 (0x10) will give a configuration file with the following format: *S17-0101-XRS-XXX-YY-10.mcs*. The maximum value is 255 (or 0xFF).

Show advanced functions: When set, this allows a user to set the Minor ID parameter in the configuration file filename. For example, a value of 31 will give a configuration file with the following name S17-0101-XRS-XXX-1F-ZZ.bin. Again, the value is formatted in decimal in the block mask and in hexadecimal in the configuration file filename. Minor ID range for RT-XSG models is from 1 to 31, or from 0x01 to 0x1F. It also enables the user to set the synchronization signal period. This value is only used for offline simulation purpose.

Inputs

This block has no inputs.

Outputs

This block has no outputs

Characteristics and Limitations

The MinorID parameter should be set to 1 or higher. 0 is usually reserved for standard configuration files.

Direct Feedthrough	N/A
Discrete sample time	N/A
XHP support	N/A
Work offline	N/A

Synchronization pulse train generator

Library

RT-XSG/Common

Block

Sync Generator

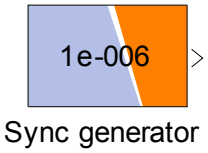

Figure 19:Sync Generator block

Mask

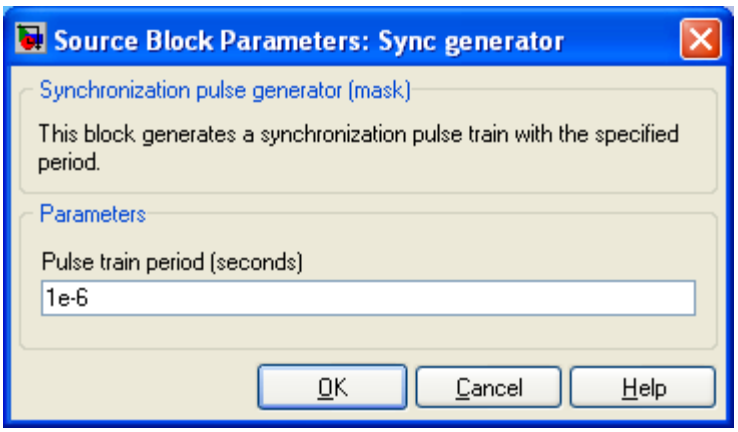

Figure 20:Sync Generator mask.

Description

This block generates a synchronization pulse train with the specified period. The width of the pulse equals the FPGA board clock period, during which its output value is set to the unsigned integer '1'. Otherwise it is set to the unsigned integer '0'.

Parameters

Period: The period of the pulse train, in seconds.

Inputs

None.

Outputs

This block has one output, corresponding to the pulse train signal. Its format is Ufix1_0.

Characteristics and Limitations

This block has no special characteristics.

Direct Feedthrough	NO
Discrete sample time	NO
XHP support	N/A
Work offline	YES

DataIN

Library

RT-XSG/Common

Block

DataIN

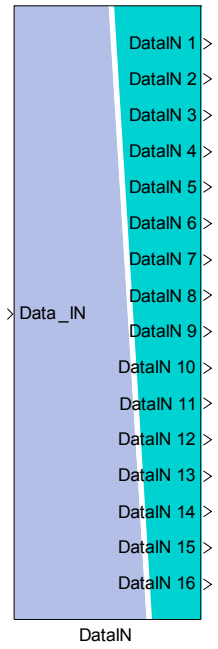

Figure 21:DataIN block

Mask

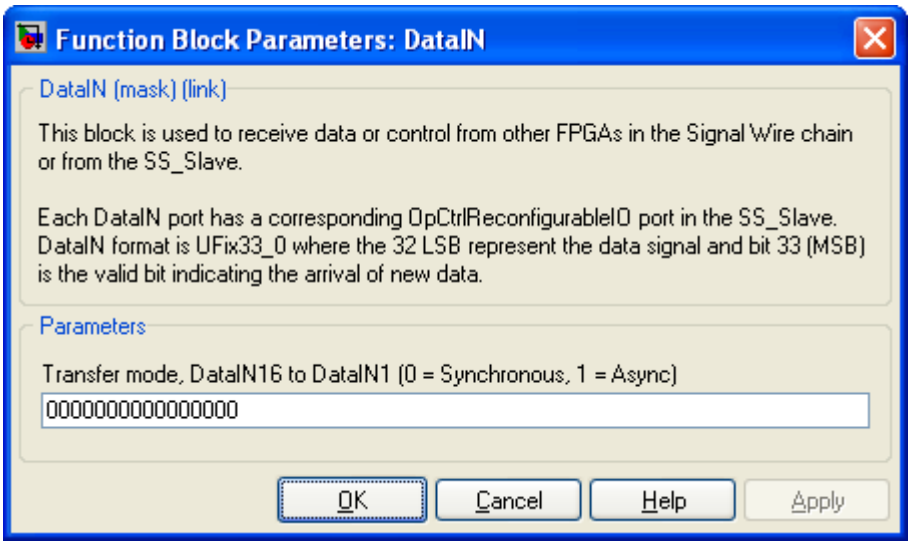
Function Block Parameters: DataIN
DataIN (mask) (link)
This block is used to receive data or control from other FPGAs in the Signal Wire chain or from the SS_Slave.
Each DataIN port has a corresponding OpCtrlReconfigurableIO port in the SS_Slave. DataIN format is UFix33_0 where the 32 LSB represent the data signal and bit 33 (MSB) is the valid bit indicating the arrival of new data.
Parameters
Transfer mode, DataIN16 to DataIN1 (0 = Synchronous, 1 = Async)
0000000000000000
OK
Cancel
Help
Apply

Figure 22:DataIN mask.

Description

This block represents the input link of the FPGA through the SignalWire bus. Data may be coming from the PC target CPU model or from a previous FPGA in a multiple chip design. Sixteen input ports are provided to the user for data samples and control signal transfers. One of the functions of this block is to perform data conversion from uint32 to the System Generator UFix33_0 data format. It is up to the user to extract the desired data out of the 32 least significant bits and to reinterpret these bits to the desired format (signed or unsigned with or without binary point).

This block is linked to the inputs of the RT-LAB OpCtrl Reconfigurable IO block found in the RT-LAB CPU model: port #1 of the OpCtrlReconfigurableIO corresponds to DataIN1, port #2 to DataIN2, etc.

Parameters

The buffering type allows a user to choose whether to synchronize incoming data, where only one data sample can be transfered per calculation step, or to the asynchronous mode, where up to 254 samples can be transfered per calculation step.

For example, a value of 010000000000101 in this field sets input ports 1 and 3 and 15 (MSB to LSB port representation) to the asynchronous mode.

Inputs

Data_IN: This is a vector of 16 uint32 type signals. Each of these signals represents an input port on the OpCtrlReconfigurableIO block of the RT-LAB CPU model.

Outputs

DataIN{1,...,16}: Each of those ports is of UFix33_0 format where the first 32 bits represent the data and bit 33 (most significant bit) is the valid signal indicating when the information is updated. When in synchronous mode (default) the valid bit is in sync with the ModelSync train pulses (active high for 10 ns). In asynchronous or in burst mode, this bit is active on arrival of the data.

Characteristics and Limitations

This block has no special characteristics.

Direct Feedthrough	NO
Discrete sample time	NO
XHP support	N/A
Work offline	YES

DataOUT

Library

RT-XSG/Common

Block

DataOUT

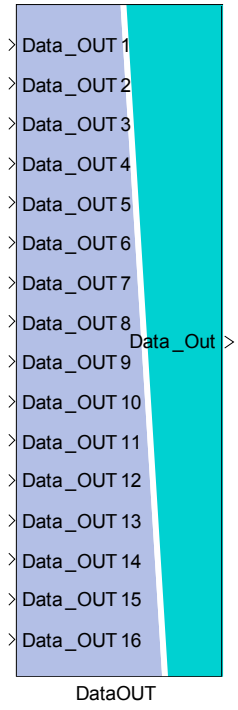

Figure 23:DataOUT block

Mask

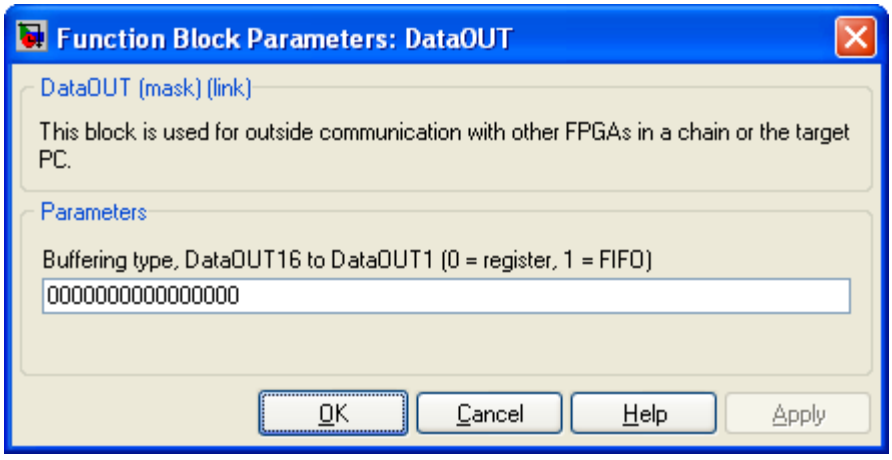
Function Block Parameters: DataOUT
DataOUT (mask) (link)
This block is used for outside communication with other FPGAs in a chain or the target PC.
Parameters
Buffering type, DataOUT16 to DataOUT1 (0 = register, 1 = FIFO)
0000000000000000
OK
Cancel
Help
Apply

Figure 24:DataOUT mask.

Description

This block represents the output link of the FPGA through the SignalWire bus. Data may be going to the PC target (RT-LAB CPU model) or to another FPGA board in a multiple chip design (only RT-LAB CPU models are supported in this version). Sixteen output ports are provided to the user for data samples and control signal transfers. One of the functions of this block is to do data conversion from the Xilinx System Generator UFix or Fix format to the uint32 data format.

This block is linked to the output ports of the OpCtrl ReconfigurableIO block found in the RT-LAB CPU model: port #1 of the OpCtrlReconfigurableIO block corresponds to DataOUT1, port #2 to DataOUT2, etc.

Parameters

The buffering type allows a user to choose whether to buffer the information in a single register where only one data sample can be transfered per calculation step or in a FIFO buffer-based mode where up to 254 samples can be transfered per calculation step.

For example, a value of 010000000000101 in this field sets a FIFO on DataOUT ports 1 and 3 and 15 (MSB to LSB port representation).

In FIFO mode, the number of samples stored is determined by the number of 10 ns pulses (one FPGA clock cycle) on bit 32 (MSB) of the port in FIFO mode per calculation step.

Inputs

Data_OUT{1,…,16}: Each of these ports is of UFix33_0 format where the first 32 bits represent the data and bit 33 (most significant bit) is the valid signal indicating when the information is updated. Bit 33 can be seen as a write signal to the buffer, whether it be a register or a FIFO, in the DataOUT block. Each of those buffers is emptied and transfered to the CPU model at the beginning of each calculation step.

Outputs

DataOUT: This is a vector of 16 signals in the uint32 format. Each one of these 16 signals represents an output port on the OpCtrlReconfigurableIO block in the RT-LAB CPU model.

Characteristics and Limitations

This block has no special characteristics.

Direct Feedthrough	NO
Discrete sample time	NO
XHP support	N/A
Work offline	YES (No for ports set in FIFO mode)

op_cosin

Library

RT-XSG/Common

Block

op_cosin

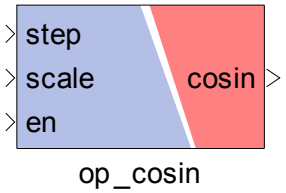

Figure 25:op_cosin block

Mask

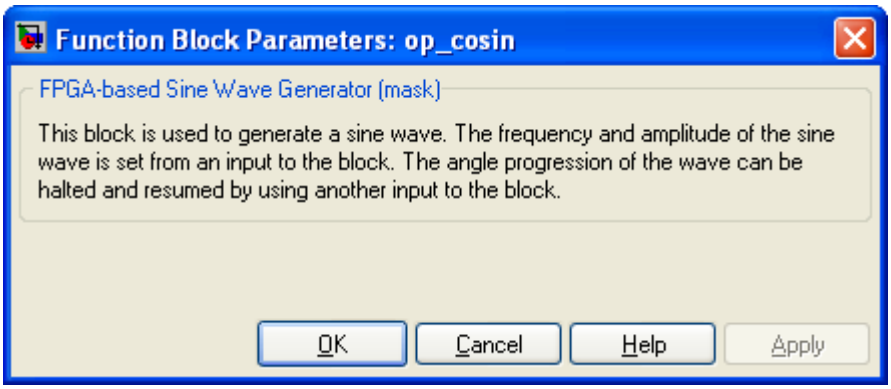
Function Block Parameters: op_cosin
FPGA-based Sine Wave Generator (mask)
This block is used to generate a sine wave. The frequency and amplitude of the sine wave is set from an input to the block. The angle progression of the wave can be halted and resumed by using another input to the block.
OK
Cancel
Help
Apply

Figure 26:op_cosin mask.

Description

This block is used to generate a sine wave. The frequency and amplitude of the sine wave is set from an input to the block. The angle progression of the wave can be halted and resumed by using another input to the block.

Parameters

None.

Inputs

Step: Angular frequency of the sine wave. This input is of a normalized UFix10_10 format, so that Step is the fraction of turns per FPGA clock cycle, included in the interval [0,1[.

Scale: Peak-to-peak amplitude of the sine wave. The format of this input is Ufix6_0.

En: This boolean input enables the angle incrementation of the sine wave generator. The output corresponds to a sine wave if the En signal is true and remains constant if En is false.

Outputs

Cosin: The generated sine wave.

Characteristics and Limitations

This block has no special characteristics.

Direct Feedthrough	NO
Discrete sample time	NO
XHP support	N/A
Work offline	YES

op_trisin

Library

RT-XSG/Common

Block

op_trisin

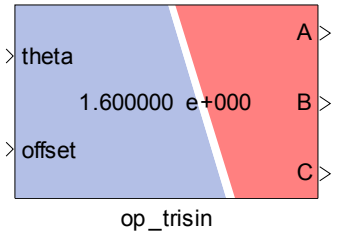

Figure 27:op_trisin block

Mask

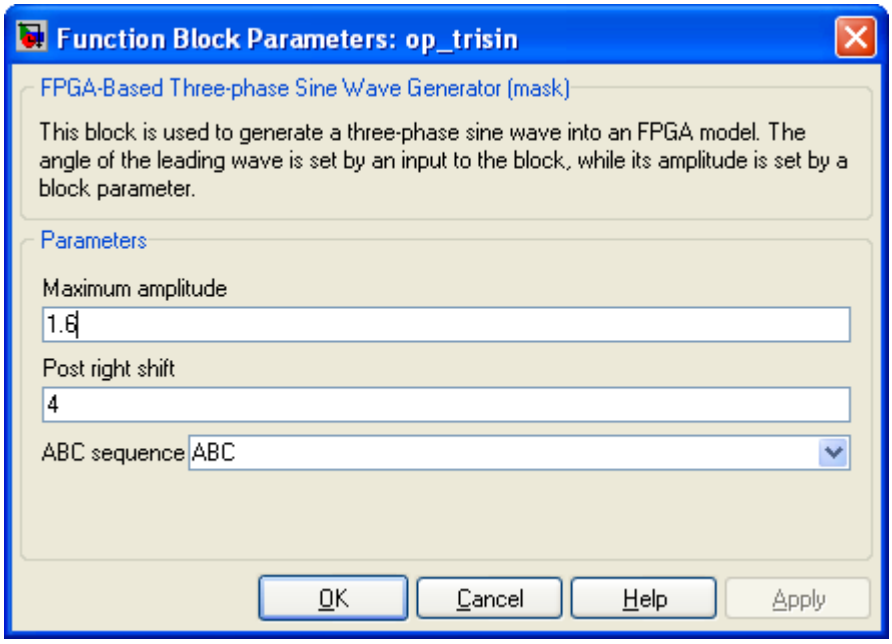
Function Block Parameters: op_trisin
FPGA-Based Three-phase Sine Wave Generator (mask)
This block is used to generate a three-phase sine wave into an FPGA model. The angle of the leading wave is set by an input to the block, while its amplitude is set by a block parameter.
Parameters
Maximum amplitude
1.6
Post right shift
4
ABC sequence
ABC
OK
Cancel
Help
Apply

Figure 28:op_trisin mask.

Description

This block is used to generate a three-phase sine wave into an FPGA model. The angle of the leading wave is set by an input to the block, while its amplitude is set by a block parameter.

Parameters

Maximum amplitude: Peak amplitude of the sine waves. The dynamic range of the wave is thus [-*Maximum amplitude*, +*Maximum amplitude*].

Post right shift: This parameter is used to pre-process the output waves to shift right the bits with the indicated integer number of positions. This is equivalent to multiplying the data by $2^{-\text{Post right shift}}$.

ABC sequence: This parameter is used to determine the order of the three phases (A=>B=>C or C=>B=>A).

Inputs

Theta: The angle of the leading sine wave, in radiants. The recommended format is Ufix10_6.

Offset: Offset of the leading sine wave. The format should be Fix10_0, and, when normalized, represents the proportion of a turn of the offset.

Outputs

A, B and C: These three outputs are the three phases of the generator.

Characteristics and Limitations

This block has no special characteristics.

Direct Feedthrough	NO
Discrete sample time	NO
XHP support	N/A
Work offline	YES

OpXsgManager

Library

RT-XSG/Tools

Block

OpXsgManager

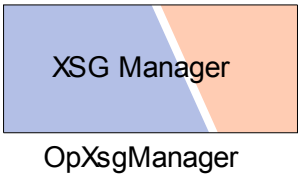

Figure 29:OpXsgManager block

Mask

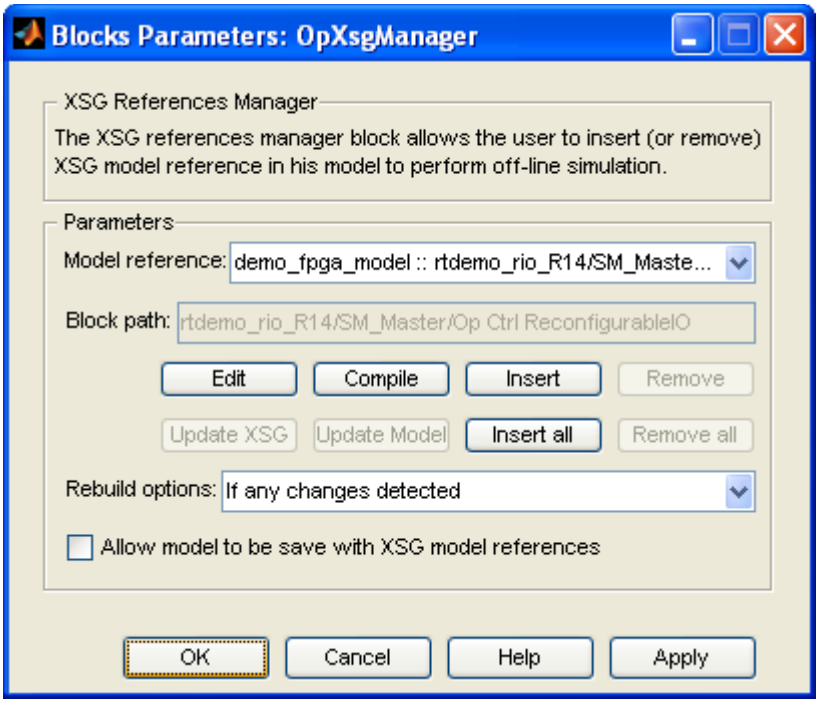

Figure 30:OpXsgManager mask.

Description

This block is in an end-of-life cycle. In future versions of RT-XSG, it will be replaced by the Opal-RT FPGA Synthesis Manager block, to be inserted directly in the RT-XSG FPGA model.

This block is the basis of the RT-XSG feature for RT-LAB. This block allows a user to create and simulate an FPGA design in a Matlab/Simulink environment through the use of the Xilinx

System Generator for DSP toolbox. The XsgManager block works in conjunction with RT-LAB Op Ctrl ReconfigurableIO blocks.

Parameters

Model Reference: This parameter shows the FPGA models associated with each of the OpCtrlReconfigurableIO blocks. This parameter corresponds to the "Xsg Model Name" parameter set in the OpCtrlReconfigurableIO blocks.

Block Path: This is the location of the OpCtrlReconfigurableIO associated with the Model reference parameter. This parameter is not editable by the user.

Edit: This button opens the RT-XSG FPGA model that is presently selected in the Model reference parameter. A user may then modify the FPGA I/O capabilities and processing functions.

Compile: Use this button to generate the FPGA configuration file corresponding to the RT-XSG model found in the reference parameter.

Rebuild Options: This parameters, used when compiling a model with RT-XSG, gives the user three choices:

- **Always**: This choice is taken into account by RT-LAB and allows a user to force the compilation of an FPGA configuration file no matter if changes where made or not to the RT-XSG FPGA model. The compile button of the XsgManager will not recompile an FPGA bitstream if there are no changes detected even though this option is set.
- **If any changes detected**: Compile the FPGA configuration file only if changes to the RT-XSG FPGA model are detected.
- **Never**: Never generate the FPGA configuration file. This avoids accidental FPGA compilations (such compilations may take tens of minutes or more).

Insert: This button will insert an xsgModel_temp subsystem underneath the OpCtrlReconfigurableIO block selected in the Model reference. This added subsystem is used for offline simulations and its contents correspond to the FPGA model associated with the OpCtrlReconfigurableIO set in its Xsg Model Name parameter. Once the FPGA model is inserted, a complete offline simulation can be performed by pressing the start simulation button in Simulink. This enables you to simulate both systems, the target node and the FPGA, exchanging data.

Insert All: Same as the "insert" button but adds an xsgModel_temp subsystem to all of the OpCtrlReconfigurableIO blocks in the model.

Remove: This button removes the xsgModel_temp subsystem selected in the Model reference parameter. Offline simulations, after the removal of the xsgModel_temp subsystem, only simulate Simulink blocks that will run on the target nodes. The OpCtrlReconfigurableIO, which represents the IO cards with the FPGA, can not be simulated. Use the Insert button for complete reconfigurable IO card and target node simulation.

Remove All: Same as Remove but removes all of the xsgModel_temp subsystems.

Update XSG: It is possible that the inserted xsgModel_temp subsystem be different of the original XSG model due to model modifications during validation an simulation. This

parameter brings the RT-XSG FPGA model up to date with the corresponding xsgModel_temp subsystem if the two are different.

Update Model: This parameter brings the xsgModel_temp subsystem up to date with the corresponding RT-XSG FPGA model if the two are different.

Allow model to be saved with XSG model references: When this option is set, it allows an RT-LAB model to be saved with "xsgModel_temp" subsystems (used for offline simulation) inserted in it. Otherwise, these subsystems must be removed before saving the model.

Inputs

None.

Outputs

None.

Characteristics and Limitations

This block has no special characteristics.

Direct Feedthrough	N/A
Discrete sample time	N/A
XHP support	N/A
Work offline	N/A

OpSGxPCManager

Library

RT-XSG/Tools

Block

OpSGxPCManager

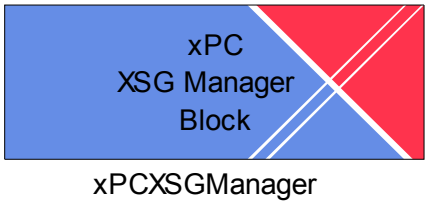

Figure 31:OpSGxPCManager block

Mask

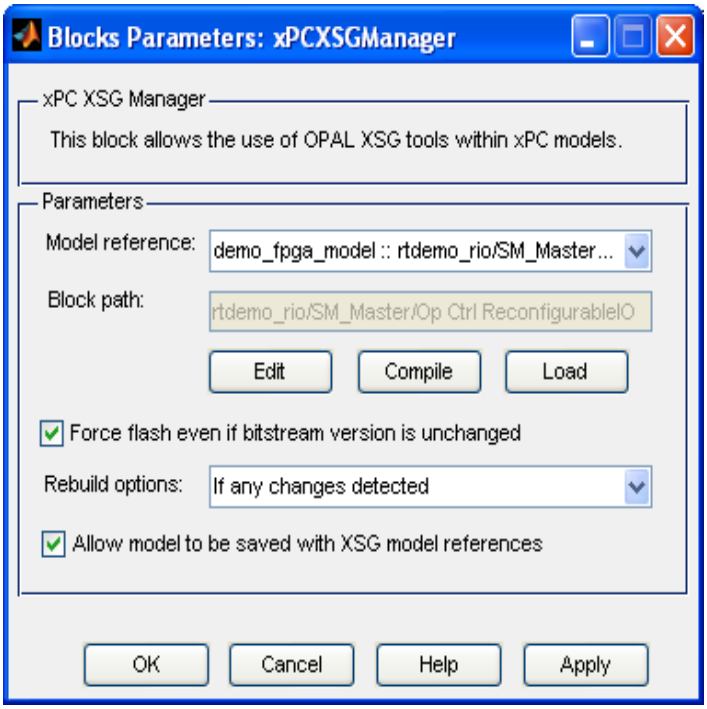

Figure 32:OpSGxPCManager mask.

Description

This block is in an end-of-life cycle. In future versions of RT-XSG, it will be replaced by the Opal-RT FPGA Synthesis Manager block, to be inserted directly in the RT-XSG FPGA model.

This block is the basis of the RT-XSG feature for xPC. This block allows a user to create and simulate an FPGA design in a Matlab/Simulink environment through the use of the Xilinx System Generator for DSP toolbox. The XsgManager block works in conjunction with a RT-LAB xPC Op Ctrl ReconfigurableIO block.

Parameters

Model Reference: This parameter shows the FPGA models associated with each of the *OpCtrlReconfigurableIO* blocks. This parameter corresponds to the Xsg Model Name parameter set in the *OpCtrlReconfigurableIO* blocks. *Note that only one OpCtrlReconfigurableIO block per model is presently supported.*

Block Path: This is the location of the *OpCtrlReconfigurableIO* associated with the Model reference parameter.

Edit: This button opens the RT-XSG FPGA model that is presently selected in the Model reference parameter. Editing the FPGA model allows the user to modify the FPGA I/O capabilities and processing functions.

Compile: Use this button to generate the FPGA configuration file corresponding to the RT-XSG model found in the reference parameter.

Rebuild Options: This parameters, used when compiling a model with RT-XSG, gives the user three choices:

- **Always**: The generation of the FPGA configuration file is started no matter if changes where made or not to the FPGA model.
- **If any changes detected**: Compile the FPGA configuration file only if changes to the FPGA model were made since the last bitstream generation.
- **Never**: FPGA configuration file generation is disabled. This avoids accidental FPGA compilations (such compilations may take tens of minutes or more) during the preparation of the CPU model.

Force flash even if bitstream version is unchanged: This option is used to force the configuration of the FPGA device on the reconfigurable board. By default programmation is disabled if the bitstream already programmed on the reconfigurable IO card has the same version identification numbers (retrieved from the Version block at the top-level of the FPGA model) as the latest bitstream generated.

Allow model to be saved with XSG model references: When this option is set, it allows an RT-LAB model to be saved with "xsgModel_temp" subsystems (used for offline simulation) inserted in it. Otherwise, these subsystems must be removed before saving the model.

Inputs

None.

Outputs

None.

Characteristics and Limitations

This block has no special characteristics.

Direct Feedthrough	N/A
Discrete sample time	N/A
XHP support	N/A
Work offline	N/A

OpXSGscope

Library

RT-XSG/Tools

Block

OpXSGscope

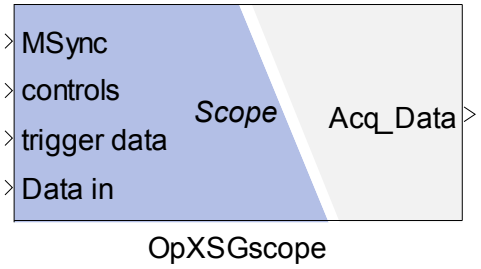

Figure 33:OpXSGscope block

Mask

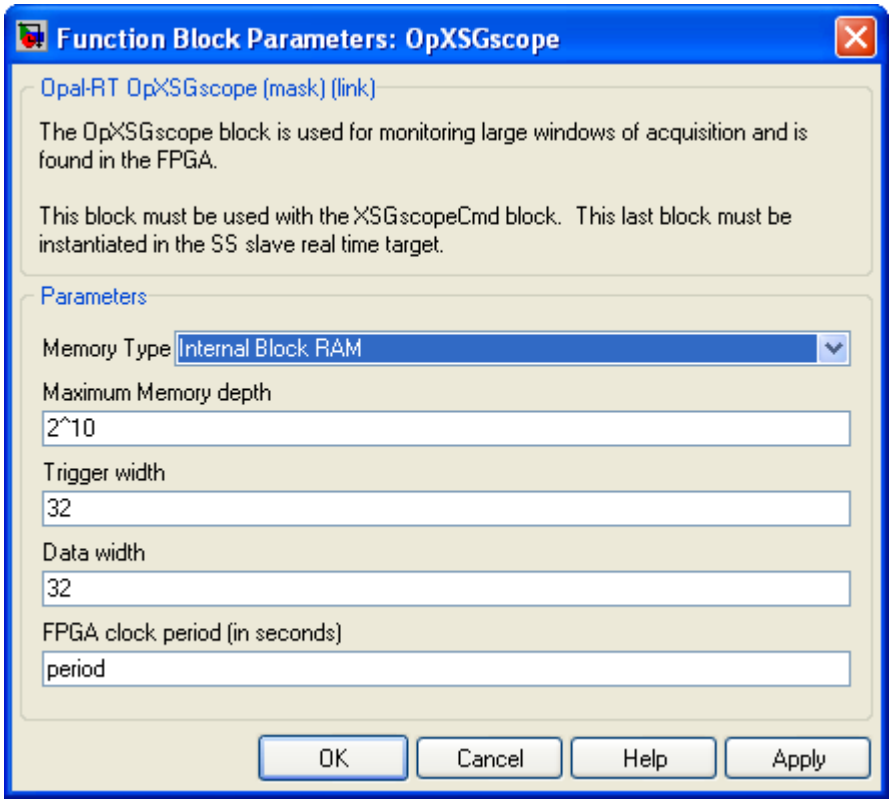

Figure 34:OpXSGscope mask.

Description

The OpXSGScope feature is not available in the current version of RT-XSG. Opal-RT Technologies is now working hard to make it available soon. In the meantime, it is proposed to use the "Chipscope" block, available from the System Generator for DSP block set. Thank you for your patience.

The OpXSGscope is a tool that enables the capture of large windows of data on an FPGA with high sampling rates of up to 100 million 32 bit data samples per second (100Msps). This data can be sent back to the console for analysis at a must slower rate. This block is to be instantiated in the FPGA and used with the OpXSGscopeCmd block that will be instantiated in the real time target.

Parameters

Memory type: Selectable type of memory for sample storage. Internal Block RAM uses the FPGA's resources for buffering. External uses the onboard SRAM memory and allows up to 2^{19} 32 bit samples to be stored.

Maximum memory depth: This parameter determines the amount of resources taken for the storage of the data samples. Memory depth is shown only if internal block RAM is chosen in the previous parameter. The maximum depth, if internal memory is chosen, is implementation dependent with a maximum of 33 block RAM in the case of an OP5130 card or 16k of 32 bit data. If external memory is chosen, the maximum depth is by default 512k x 32 samples.

Trigger width: Width of the input trigger port. Maximum of 32 bits.

Data width: Width of the input port Data in. This value as an incidence on the maximum sample rate. The maximum sample rate is of 100MHz if data width is set to 32 bits or less. Higher values diminish by a factor x the maximum sample rate, where x is calculated as follows: x = ceil(Data width/32). For example, if the maximum width of 128 bits is set, the maximum sampling rate will fall to 25Msps. The sampling rate is set in the OpXsgScopeCmd block found in the real time target.

Inputs

Model Sync: Calculation step. Connect to ModelSync "From" block for synchronicity with the real time target computer.

Controls: This port is to be connected to the DataIN block in order for the XsgScope to receive its controls from the OpXSGscopeCmd block found in the real time target.

Trigger data: This is the data that will be compared with the trigger value set by the OpXSGscopeCmd block found in the real time target. Only 32 bits wide for this version.

Data in: Data to be stored in the memory on trigger condition. Up to 128 bits wide.

Outputs

Acq data: Data sent to the real time target for analysis. This port is connected to the DataOUT block. The corresponding port on the DataOUT block must be set in buffer mode. Since the information coming out of this ports is in bursts (more than one sample per calculation step) it must be connected to a mux block in the real time target in order to extract the samples in vector form where the first vector contains the status of the buffer on the FPGA and the succeding vectors are the data samples.

The Status vector is composed of an acquisition buffer full indicator (first bit) and buffer empty (second bit).

Characteristics and Limitations

The trigger width is limited to 32 bits in this version.

Direct Feedthrough	N/A
Discrete sample time	N/A
XHP support	N/A
Work offline	NO

XSGscopeCmd

Library

RT-XSG/Tools

Block

XSGScopeCmd

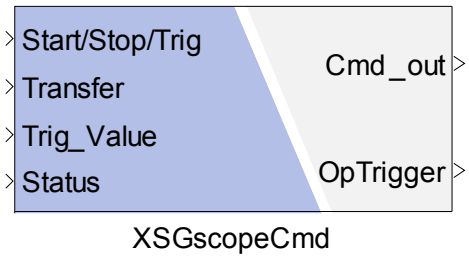

Figure 35:XSGScopeCmd block

Mask

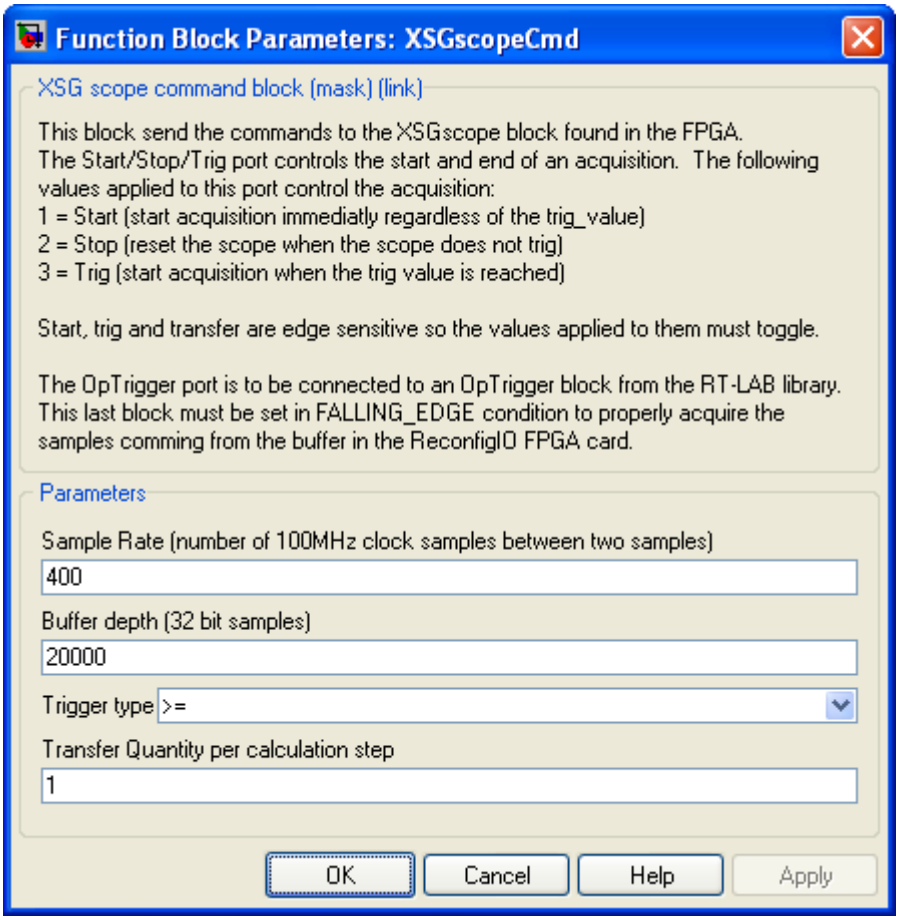
Function Block Parameters: XSGscopeCmd
XSG scope command block (mask) (link)
This block send the commands to the XSGscope block found in the FPGA.
The Start/Stop/Trig port controls the start and end of an acquisition. The following values applied to this port control the acquisition:
1 = Start (start acquisition immediatly regardless of the trig_value)
2 = Stop (reset the scope when the scope does not trig)
3 = Trig (start acquisition when the trig value is reached)
Start, trig and transfer are edge sensitive so the values applied to them must toggle.
The OpTrigger port is to be connected to an OpTrigger block from the RT-LAB library. This last block must be set in FALLING_EDGE condition to properly acquire the samples comming from the buffer in the ReconfigIO FPGA card.
Parameters
Sample Rate (number of 100MHz clock samples between two samples)
400
Buffer depth (32 bit samples)
20000
Trigger type
>=
Transfer Quantity per calculation step
1
OK
Cancel
Help
Apply

Figure 36:XSGScopeCmd mask.

Description

The OpXSGScope feature is not available in the current version of RT-XSG. Opal-RT Technologies is now working hard to make it available soon. In the meantime, it is proposed to use the "Chipscope" block, available from the System Generator for DSP block set. Thank you for your patience.

The OpXSGscopeCmd block must be instantiated in the real time target and is used to control the hardware OpXsgScope block found in the FPGA.

Parameters

Sample rate : Enter the number of FPGA clock cycles (or the number of 10 ns cycles) between the capture of two data samples. The minimum is one clock cycle and the maximum is 2^{28} cycles between two samples.

Buffer depth: This is the amount of 32 bit data samples that will be stored in the memory buffers on the FPGA once the trigger condition is encountered. Once this depth is reached in the memory, it is ready to be sent back to the computer target by setting the transfer input port to 1. The maximum depth is determined by the type of memory chosen on the FPGA. In the case of the external SRAM memory, the maximum is 2^{19} or 524288 32-bit data samples.

Trigger type: This is the type of condition that will trigger the sampling. Once this condition is encountered, the memory is filled up to the buffer depth set above.

Transfer quantity per calculation step: The OpXsgScope found on the FPGA is a store and forward engine. It can aquire information at a much higher rate the calculation step of the real time target. When the memory buffers on the FPGA card are full, they are transfered back to the target at a rate that is configurable by this parameter from one 32 bit data sample per calculation step up to 253. The higher this number the more the computer target will require bandwidth to process this information but the faster it will be to transfer the information. For example, when using the external SRAM memory on the FPGA and setting the buffer depth at 500000 and the transfer rate at 1, it can take almost one minute to transfer all these samples if the calculation step of the model is set at 100µs.

Inputs

Start/Stop/Trig: This command starts the acquisiton. Applying a value of 1 to this port starts the acquisition on the FPGA immediatly regardless of the trigger value. Applying a value of 2 stop the acquisition and a value of 3 sets the scope in trigger mode where data samples will be written in the memory buffer only when the trigger conditions are encountered.

Transfer: This port is given to allow control on the transfer of the acquired data. Apply a value of 1 to this port to start transfer.

Trig_value: This is the value that will be compared with the data presented on the OpXSGscope Trigger data port in the FPGA.

Status: The acquired data coming from the FPGA is in vector form where the first vector is the status of the buffer in the FPGA and this vector must be connected to the status port. The information is available at every calculation step. The number of vectors coming out of the ReconfigIO port containing the acquired data depends on the **Transfer quantity per calculation step** parameter. If this last parameter is set to 0, there will be only one vector available per calculation step, the status. If the transfer quantity parameter is set to one,

there will then be two vectors where the first one is again the status and the second vector is the 32 bit data sample and so on.

See the *OpXsgScope* block documentation for the signification of the status bits.

Outputs

Cmd out : The parameters set in the mask and the input ports of the OpXsgScopeCmd block are sent to the harware version of the scope in the FPGA through this port. The output is already in uint32 format and can be directly connected to the ReconfigIO block that will send these command to the FPGA.

OpTrigger: This port toggles from high to low when the first requested data samples arrive from the FPGA. It can be directly connected to an RT-LAB *OpTrigger* block whose Condition parameter is set to FALLING_EDGE. The *OpTrigger* block can in turn be connected to an RT-LAB *OpWriteFile* block for storing large amounts of data.

Characteristics and Limitations

Direct Feedthrough	N/A
Discrete sample time	N/A
XHP support	N/A
Work offline	NO